Probabilistic Models for Music

THÈSE N^{O} 4148 (2008)

PRÉSENTÉE LE 28 JUILLET 2008

À LA FACULTE SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE DE L'IDIAP

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Jean-François PAIEMENT

M.Sc. en informatique, Université de Montréal, Québec, Canada
et de nationalité canadienne

acceptée sur proposition du jury:

Prof. P. Vandergheynst, président du jury
Prof. H. Bourlard, Dr S. Bengio, directeurs de thèse
Prof. J. M. Iñesta Quereda, rapporteur
Prof. J. Larsen, rapporteur
Prof. J.-Ph. Thiran, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL
2008

Résumé

Dans cette thèse, nous proposons une analyse des données musicales symboliques d'un point de vue statistique, en utilisant des techniques modernes d'apprentissage machine. L'argument principal de cette thèse consiste à montrer qu'il est possible de concevoir des *modèles génératifs*, qui sont en mesure de *prédire* et de *générer* des données musicales, dans un style similaire à celui d'un corpus d'entraînement, et ce en utilisant un minimum de données.

Nos contributions majeures dans cette thèse sont de trois ordres:

- Nous avons montré empiriquement que des dépendances probabilistes à long terme sont présentes dans les données musicales. Nous rapportons des mesures quantitatives de telles dépendances;
- Nous avons montré empiriquement qu'utiliser des connaissances spécifiques au domaine musical permet de mieux modéliser les dépendances à long terme au sein des données musicales qu'avec des modèles de séries temporelles standards. Ainsi, nous définissons plusieurs modèles probabilistes destinés à capturer divers aspects des données musicales polyphoniques. Ces modèles peuvent aussi être échantillonnés afin de générer des séquences musicales réalistes;
- Nous avons conçu diverses représentations musicales pouvant être utilisées directement comme observations par les modèles probabilistes proposés.

Mots-clés: Apprentissage machine, musique, modèles probabilistes, modèles génératifs, progressions d'accords, mélodies.

Abstract

This thesis proposes to analyse symbolic musical data under a statistical viewpoint, using state-of-the-art machine learning techniques. Our main argument is to show that it is possible to design *generative models* that are able to *predict* and to *generate* music given arbitrary contexts in a genre similar to a training corpus, using a minimal amount of data. For instance, a carefully designed generative model could guess what would be a good accompaniment for a given melody. Conversely, we propose generative models in this thesis that can be sampled to generate realistic melodies given harmonic context.

Most computer music research has been devoted so far to the direct modeling of audio data. However, most of the music models today *do not* consider the musical structure at all. We argue that reliable symbolic music models such a the ones presented in this thesis could dramatically improve the performance of audio algorithms applied in more general contexts.

Hence, our main contributions in this thesis are three-fold:

- We have shown empirically that long term dependencies *are* present in music data and we provide quantitative measures of such dependencies;

- We have shown empirically that using domain knowledge allows to capture long term dependencies in music signal better than with standard statistical models for temporal data. We describe many probabilistic models aimed to capture various aspects of symbolic polyphonic music. Such models can be used for music prediction. Moreover, these models can be sampled to generate realistic music sequences;

- We designed various representations for music that could be used as observations by the proposed probabilistic models.

Keywords: machine learning, music, probabilistic models, generative models, chord progressions, melodies.

Acknowledgments

Cette thèse est dédiée à Josiane, le plus beau coeur du monde.

J'aimerais adresser mes remerciements les plus sincères à Samy Bengio. Les quatres dernières années furent pour moi vraiment enrichissantes et agréables, et ce fût en grande partie grâce à lui. Samy a toujours été là au bon moment pour me guider ou m'encourager, et je lui en suis vivement reconnaissant.

J'aimerais aussi remercier Hervé Bourlard, grâce à qui l'IDIAP est un endroit si propice pour faire de la recherche scientifique de qualité.

Thanks to Douglas Eck, who provided me a wonderful research environment for more than a year in Université de Montréal.

Finally, I wish to thank all the agencies that made this research possible. The work reported in this thesis was supported in part by the IST Program of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, funded in part by the Swiss Federal Office for Education and Science (OFES) and the Swiss NSF through the NCCR on IM2.

Contents

1 Introduction

1.1 Statistical Modeling of Music

Most people – and even young children – agree when deciding whether a sequence of notes can be considered as a melody or not. People also mostly agree when asserting if a melody is appropriate in a given musical context. Moreover, while there is no solid ground truth about musical genres, most people agree when discriminating between heavy metal and reggae, even though the same musical instruments are used when playing these two musical genres.

What intrinsic properties a sequence of notes should possess to be a melody? What relations this sequence of notes should have with other musical components to be considered valid in a specific context? What makes genre A different from genre B?

When telling that a particular song genre is rock music, the listener identifies musical patterns that are common to most other rock songs, under some invariances. However, all rock songs are far from being all identical. Where and how is it possible to put some variability in a rock song? Where is there specific rhythmic or melodic patterns that *should* be present in a song so that it can be considered as rock music?

This thesis proposes to explore these questions under a statistical viewpoint, using state-of-the-art machine learning techniques. The main argument of this thesis is to show that it is possible to design *probabilistic models* that are able to *predict* and to *generate* music given arbitrary contexts in a genre similar to the training corpus, using a minimal amount of music data. The exact meaning and the realm of the last sentence will become clear while reading this chapter, where we first introduce basic musical concepts required for the reader to understand the material in this thesis. Then, we introduce some elements of *machine learning*, which is the design of algorithms that can learn

from datasets of examples.

Musical events are tied by very long term statistical dependencies. This has proved very difficult to model with traditional statistical methods. The problem of long-term dependencies is not limited to music, nor to one particular probabilistic model [Bengio et al., 1994]. One of the main contributions of this thesis is to show that statistical dependencies in music usually follow standard patterns that can be effectively captured with carefully designed models.

1.2 The Nature of Music

While we do not expect the reader to be an expert in music, some basic elements of music theory [Sadie, 1980] are necessary to understand the models presented in this thesis.

Musical notes can be described by 5 distinct characteristics: namely pitch, rhythm, loudness, spatialization, and timbre.

1.2.1 Pitch

Pitch is the perceived frequency of a sound [Krumhansl, 1979]. The frequency content of an idealized musical note is composed of a fundamental frequency and integer multiples of that frequency. Human pitch perception is logarithmic with respect to fundamental frequency. Thus, we normally refer to the pitch of a note using *pitch classes*. In English, a pitch class is defined by a letter. For instance, the note with the fundamental frequency of 440 Hz is called A. In the Western music culture, the *chromatic scale* is the most common method of organizing notes. When using the *equal temperament*, each successive note is separated by a *semi-tone*. Two notes separated by a semi-tone have a fundamental frequency ratio of the twelfth root of two (approximately 1.05946). Using this system, the fundamental frequency is doubled every 12 semi-tones. The *interval* between two notes refers to the space between these two notes with regard to pitch. Two notes separated by 12 semi-tones are said to be separated by an octave, and have the same pitch-class. For instance, the note with fundamental frequency at 880 Hz is called A, one octave higher than the note A with the fundamental frequency at 440 Hz. We say that the *interval* between these two notes is an octave.

The symbol # (sharp) raises a note by one semi-tone. Conversely, the symbol b (flat) lowers a note by one semi-tone. Most of the pitch classes are separated by one tone (i.e. two semi-tones), except for notes E and F, as well as B and C, that are separated only by one semi-tone. Table 1.1 shows

Table 1.1. Fundamental frequencies for pitch-classes ranging from A 440 Hz to A 880 Hz. There is only one semi-tone between E and F, as well as between B and C.

Pitch class	Fundamental frequency (Hz)
A	440
A# / Bb	466.16
B	493.88
C	523.25
C# / Db	554.37
D	587.33
D# / Eb	622.25
E	659.26
F	698.46
F# / Gb	739.99
G	783.99
G# / Ab	830.61
A	880

fundamental frequencies for pitch-classes ranging from A 440 Hz to A 880 Hz. In this system, A# and Bb refer to the same note. Two pitch classes that refer to the same pitch are called *enharmonics*. In this thesis, we consider enharmonics to be completely equivalent.

1.2.2 Rhythm

In most music notations, rhythm is defined relatively to an underlying beat that divides time in equal parts. The speed of the beat is called the *tempo*. For instance, when the tempo is 120, we count 120 beats per minute (BPM), or two beats per second. *Meter* is the sense of strong and weak beats that arises from the interaction among hierarchical levels of sequences having nested periodic components. Such a hierarchy is implied in Western music notation, where different levels are indicated by kinds of notes (whole notes, half notes, quarter notes, etc.) and where bars (or alternatively *measures*) establish segments containing an equal number of beats [Handel, 1993]. Kinds of notes are defined relatively to each other. Whole notes have always twice the length of half notes, which have twice the length of quarter notes, and so on. The number of beats

per bar is usually defined in the beginning of a song by the *time signature*. Also, depending on the meter definition, kinds of notes can last for variable number of beats. For instance, in most four-beat meters, a quarter note lasts one beat. Hence, an eight note lasts for half a beat, a half note lasts for two beats, and a whole note lasts for four beats. If the tempo is 120, we play one half note per second and there is two half notes per bar.

When some beats in the measure are played consistently stronger or weaker than others, a recognizable *metrical grid* is established. Different metrical grids are possible for the same meter, and help provide a temporal framework around which a piece of music is organized. For example, while virtually all modern pop songs are built around the same four-beat meter, different metrical grids yield different musical styles. For instance, a metrical grid which stresses beats 2 and 4 is a key component of the classic 1950s rock-and-roll.

1.2.3 Loudness, Spatialization, and Timbre

The loudness of a sound depends on the amplitude of its waveform. Traditional music notation defines loudness using Italian qualitative terms such as forte (loud) and piano (soft).

Spatialization is just the position in space from which the sound originates. Traditional music notation used to ignore this aspect of sounds. Nowadays, most music composers use spatialization as an important part of their musical language.

Timbre is the most complicated and less understood aspect of a sound [Schouten, 1968]. It could be simply defined as being all the variable aspects of a sound that are neither pitch, rhythm, loudness, nor spatialization. More intuitively, the timbre of a sound is what makes the difference between two notes played at the same pitch, same rhythm, and same loudness with two *different* musical instruments. An important feature that determines timbre is the relative amplitude of each of the harmonics of a musical sound. For instance, an organ pipe reinforces the odd harmonics. Hence, a synthesized waveform with very loud odd harmonics compared to even harmonics is likely to sound like an organ.

While loudness, spatialization, and timbre are really important aspects of sound, this thesis is mostly concerned with the modeling of pitch and rhythm.

1.2.4 Tonal Music

Tonal music comprises most of the Western music that has been written since J.-S. Bach (including contemporary pop music.) One of the main features of

tonal music is its organization around *chord progressions*. A chord is a group of three or more notes (generally five or less.) A chord progression is simply a sequence of chords. In general, the chord progression itself is not played directly in a given musical composition. Instead, notes comprising the current chord act as central polarities for the choice of notes at a given moment in a musical piece. Given that a particular temporal region in a musical piece is associated with a certain chord, notes comprising that chord or sharing some harmonics with notes of that chord are more likely to be present. In probabilistic terms, the current chord can be seen as a latent variable (local in time) that conditions the probabilities of choosing particular notes in other music components, such as melodies or accompaniments. In typical tonal music, most chord progressions are repeated in a cyclic fashion as the piece unfolds, with each chord having in general a length equal to integer multiples of the shortest chord length. Also, chord changes tend to align with bars. Since chord changes usually occur at fixed time intervals, they should be much simpler to detect in audio signal than beginnings and endings of musical notes, which can happen almost everywhere.

Meter, rhythm, and chord progressions provide a framework for developing musical melody. For instance, in most contemporary pop songs, the first and third beats are usually emphasized. In terms of melodic structure, this indicates that notes perceptually closer to the chord progression are more likely to be played on these beats while more "dissonant" notes can be played on weaker beats. For a complete treatment of the role of meter in musical structure see Cooper and Meyer [1960].

In most tonal music theories, chord names are defined by a root note that can either be expressed by its absolute pitch-class or by its relation with the current key. The key is the quality by which all the notes of a song are considered in relation with a central tone, called the tonic. The key of a song is designated by a note name (the tonic), and is the base of a musical scale from which most of the notes of the piece are drawn. Most commonly, that scale can be either in major or minor mode. See for instance Schmeling [2005] for a thorough introduction to musical scales.

Throughout this thesis, we define chords by giving the pitch class letter, sometimes followed by symbol `#` (sharp) to raise a given pitch class by one semi-tone. Finally, each pitch class is followed by a digit representing the actual octave where the note is played. For instance, the symbol `c1e2a#2d3` stands for the 4-note chord with a `c` on the first octave, an `e` and an `a` sharp (`b` flat) on the second octave, and finally a `d` on the third octave. The third octave is the octave that contains A 440 Hz.

1.2.5 Computer Applications in Music

We can divide the spectrum of computer applications that deal with music into two overlapping categories: Applications concerned directly with audio data and applications that deal with more abstract representations of music data.

Audio data

With the widespread use of portable music players, most computer users today have to deal with large digital music database. Plenty of software tools (e.g. iTunes) are available to play, retrieve, and store huge quantities of audio files. A lot of research has been done in the recent years in the area of music information retrieval [Pachet and Zils, 2003; Berenzweig et al., 2003; Slaney, 2002; Peeters and Rodet, 2002; Pachet et al., 2001]. Abstract features can be extracted from audio signal as a preprocessing step for various applications [Burges et al., 2003; Foote, 1997; Scheirer, 2000; Davy and Godsill, 2003].

For instance, many algorithms have been proposed to do automatic musical genre recognition [Zanon and Widmer, 2003; Aucouturier and Pachet, 2003; Tzanetakis et al., 2001; Tzanetakis and Cook, 2002; Pachet and Cazaly, 2000; Soltau et al., 1998]. This application is really interesting and has obvious huge commercial interest. An online music store would like to be able to suggest music similar to what a consumer already bought. However, music genre recognition suffers from two major drawbacks. First, most human similarity judgments about music similarity are *not* based on audio data itself, but on other meta-informations surrounding the audio file, such as the identity of the artist, its popularity, ethnicity, the year of production, the country of origin, and many other factors. All this information cannot be extracted from audio data. For instance, some chord progressions in Beatles songs are almost similar to chord progressions in popular music of the 17th century, while these two musical genres are considered completely different by most people. Another fundamental problem in music genre recognition is that nobody agree on genre taxonomy, hence there is no ground truth for algorithm evaluation. A promising approach would be to allow the users to define their own taxonomy and let the learning algorithms train on these customized musical genres.

MIDI stands for Musical Instrument Digital Interface, an industry-standard interface used on electronic musical keyboards and PCs for computer control of musical instruments and devices. An interesting challenge arising in the music information retrieval context is transcription, i.e. converting audio data into

any kind of symbolic representation such as MIDI or traditional music notation [Cemgil et al., 2003, 2006; Klapuri, 2001; Walmsley, 2000; Sterian, 1999; Martin, 1996]. However, because of fundamental difficulties inherent to the nature of sound, state-of-the-art techniques are not good enough for most practical applications. Suppose for instance that the even harmonics of a particular note have very high relative amplitude. The same sound could be produced when playing two notes separated by one octave, if the timbre if different. These difficulties have been addressed in pitch tracking algorithms [Saul et al., 2003; de Cheveigné and Kawahara, 2002; Parra and Jain, 2001; Klapuri et al., 2000; Klapuri, 1999; Slaney and Lyon, 1990] with mixed success. Pitch tracking is a sub-problem of transcription. An algorithm for pitch tracking tries to detect fundamental frequencies of notes in audio signals without regard to rhythm representation.

Algorithms for beat tracking [Tzanetakis et al., 2002; Goto and Muraoka, 1998; Cemgil et al., 2000; Goto, 2001; Scheirer, 1998] are more successful than transcription algorithms. Such algorithms can be used as preprocessors for alignment before actual transcription. Moreover, accurate beat tracking can be useful to synchronize visual effects with music in real-time.

Symbolic data

All the applications described so far directly consider audio data. Because of the sudden huge popularity of portable audio players, research about algorithms that sort, retrieve, and suggest music have become really important recently. However, state-of-the-art transcription algorithms are not reliable today. Hence, most of the music models today *do not* consider the musical structure at all. They mostly rely on local properties of audio signal, such as texture, or short term frequency analysis. For instance, in most current approaches for transcription or pitch tracking, the algorithms have to rely on strong assumptions about timbre or the number of simultaneous notes to decide how many notes are simultaneously played, and to identify these notes. The general applicability of these algorithms is thus limited.

An interesting intermediate approach in the context of genre classification is proposed by Lidy et al. [2007] to overcome this problem. In this work, audio features and symbolic features are combined, which leads to better classification results. Symbolic features are extracted from audio data through an intermediate transcription step. Hence, while transcription performances are far from being perfect by themselves, they appear to be sufficient to provide worthwhile information for genre classification purposes.

However, very little research has been done to model symbolic music data compared to the important efforts deployed to model audio data. Accurate symbolic music models such as the ones presented in this thesis could dramatically improve the performance of transcription algorithms applied in more general contexts. They would provide "musical knowledge" to algorithms that currently only rely on basic sound properties to take decisions. In the same way, natural language models are commonly used in speech transcription algorithms [Rabiner and Schafer, 1978]. As a simple example, suppose that a transcription algorithm knows the key of a particular song and tries to guess the last note of a song. The prior probability that this note would be the tonic would be very high, since most of the songs in any corpus end on the tonic.

Another advantage of symbolic music data is that it is much more compressed than audio data. For instance, the symbolic representation of an audio file of dozens of megabytes can be just a few kilobytes large. These few kilobytes contain most of the information that is needed to reconstruct the original audio file. Thus, we can concentrate on essential psychoacoustic features of the signal when designing algorithms to capture long term dependencies in symbolic music data.

Finally, the most interesting advantage of dealing directly with symbolic data is the possibility of designing realistic music generation algorithms. Most of the probabilistic models presented in this thesis are generative models (c.f. Section 1.3.1). Thus, these models can be sampled to generate genuine musical events, given other musical components or not. For instance, we introduce in Chapter 4 an algorithm that generates the most probable sequence of chords given any melody, following the musical genre of the corpus on which the algorithm was trained.

State-of-the-art research papers in the area of symbolic music modeling are described in the chapters of this thesis related to corresponding applications.

1.3 Elements of Machine Learning

Artificial intelligence [Russell and Norvig, 2002] is a well-known subfield of computer science, which is concerned with producing machines to automate tasks requiring "intelligent" behavior. Early artificial intelligence systems were usually based on the definition of a set of rules. These rules were used by computers to solve problems, make decisions, or take actions in response to some inputs coming from the real world. However, the sets of rules required to solve complicated tasks such as natural language understanding or visual object recognition turned out to be much too complicated to design and encode for

practical purposes [Duda et al., 2000a]. Such systems were lacking flexibility for further improvement and required huge amounts of human effort to encode domain knowledge.

To overcome these limitations, machine learning emerged [Rosenblatt, 1958; Vapnik, 1998] as a subfield of artificial intelligence concerned with the design of algorithms that can *learn* from examples. Since datasets of examples are usually very large in practice, the domain of machine learning is very close to statistics. Excellent introductions to the elements of machine learning can be found in Bishop [2006] and Duda et al. [2000b].

We assume in this thesis that the reader is familiar with the basic concepts of random variables and probability distributions [Billingsley, 1995; Breiman, 1968; Feller, 1971]. Machine learning models can be divided into two main categories, namely *discriminative* models and *generative* models. As a simple application of machine learning, let us consider the problem of classification. Let $\mathbf{x}$ be a multidimensional random variable corresponding to attributes of objects that we observe in a dataset. Let y be a discrete random variable where each state correspond to a class of the objects in the datasets. For instance, the observed $\mathbf{x}$ vectors may correspond to pixels in images while the values of y would correspond to the identities of these objects. Discriminative models try to estimate the distribution $p(y|\mathbf{x})$. In other words, a discriminative model concentrates on a particular task, with less emphasis on the distribution $p(\mathbf{x})$ of the dataset.

Instead, the models presented in this thesis belong to the category of generative models. *Bayes rule* provides that

$$p(y|\mathbf{x}) \propto p(\mathbf{x}|y)p(\mathbf{x}) \ . \tag{1.1}$$

Generative approaches aim to model the right part of Eq. (1.1) as an intermediate task. This is usually much more difficult since it requires to model the joint distribution of all the components in $\mathbf{x}$. This is especially hard when $\mathbf{x}$ has high dimension, or when considering datasets with limited size. In practice, discriminative models are more efficient when the goal is to accomplish a specific task. On the other hand, reliable generative models are more powerful when many tasks are to be solved by the same model or when missing values are present in datasets. Generative models can also be sampled to generate new instances of data.

In very general terms, discriminative modeling may be seen as *engineering* (i.e. solving a particular task in the most efficient way possible). On the other hand, generative modeling may be closer to *science* (i.e. understanding the fundamental principles and the relationships between empirical observations).

1.3.1 Generative Models

The advantages of generative models for music are plenty: One can sample the learned distributions to generate new music. Moreover, one can use the learned distributions to infer the probability of musical observations given other music components. For instance, a generative model can guess what would be a good accompaniment for a given melody, as we present in Chapter 4. Conversely, one could sample a generative model to generate realistic melodies given harmonic context, as is shown in Chapter 6.

As pointed out in Section 1.2.5, most applications dealing with symbolic music data take as inputs some musical components (e.g. melodies, chord progressions, or audio excerpts) and produce some other musical components. Modeling the nature of the relationships between these musical components appears to be the common ground to all these applications. Generative models provides an ideal framework for such a modeling task.

The probabilistic models used in this thesis are described using the graphical model framework. Graphical models [Lauritzen, 1996] are useful to define probability distributions where graphs are used as representations for a particular factorization of joint probabilities. Vertices are associated with random variables. A directed edge going from the vertex associated with variable A to the one corresponding to variable B accounts for the presence of the term $P(B|A)$ in the factorization of the joint distribution of all the variables in the model. For instance, the graphical model in Figure 1.1 means that the joint probability of variables A, B, and C (i.e. $P(A, B, C)$) can be expressed as $P(A)P(B|A)P(C|A)$. Defining a graphical model representation for a set of random variables amounts to defining a set of *independence assumptions* between these variables, by factorization of their joint distribution. The process of calculating probability distributions for a subset of the variables of the model given the joint distribution of all the variables is called *marginalization* (e.g. deriving $P(A, B)$ from $P(A, B, C)$). The graphical model framework provides efficient algorithms for marginalization and various learning algorithms can be used to learn the parameters of a model, given an appropriate dataset.

The Expectation-Maximization (EM) algorithm [Dempster et al., 1977; Bilmes, 1997] can be used to estimate the conditional probabilities of the hidden variables in a graphical model. Hidden variables are variables that are neither observed during training nor during evaluation of the models. These variables represent underlying phenomena that have an impact on the actual observations, but that cannot be observed directly. Such variables are used in probabilistic models to distribute or to compress information transmitted

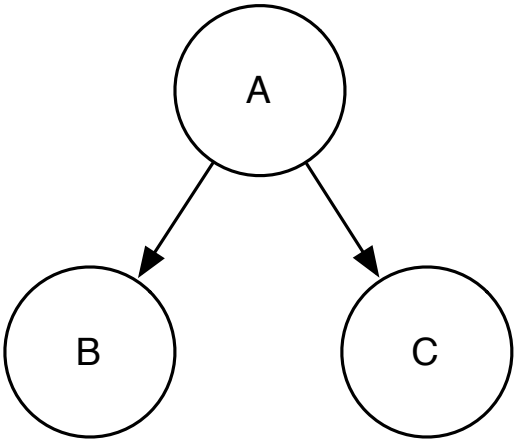

Figure 1.1. Graphical model representation of the joint probability $P(A, B, C) = P(A)P(B|A)P(C|A)$.

between observed random variables. The EM algorithm proceeds in two steps applied iteratively over a dataset until convergence of the parameters. Firstly, the E step computes the expectation of the hidden variables, given the current parameters of the model and the observations of the dataset. Secondly, the M step updates the values of the parameters in order to maximize the joint likelihood of the observations and the expected values of the hidden variables.

Marginalization must be carried out in the models proposed in this thesis both for learning (during the expectation step of the EM algorithm) and for evaluation. The inference in a graphical model can be achieved using the Junction Tree Algorithm (JTA) [Lauritzen, 1996]. In order to build the junction tree representation of the joint distribution of all the variables of the model, we start by moralizing the original graph (i.e. connecting the non-connected parents of a common child and then removing the directionality of all edges) so that some of the independence properties in the original graph are preserved. In the next step (called triangulation), we add edges to remove all chord-less cycles of length 4 or more. Then, we can form clusters with the maximal cliques of the triangulated graph. The Junction Tree representation is formed by joining these clusters together. The functions associated to each cluster of the Junction Tree are called *potential* functions. We finally apply a message passing scheme between the potential functions. These functions can be normalized to give the marginalized probabilities of the variables in each cluster. Given observed data,

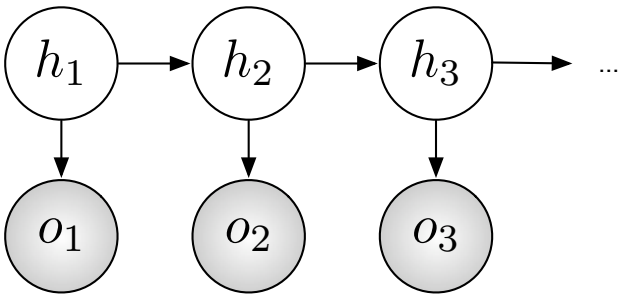

Figure 1.2. Hidden Markov Model. Each node is associated to a random variable and arrows denote conditional dependencies. When learning the parameters of the model, white nodes are hidden whereas grey nodes are observed.

the properties of the Junction Tree allow the potential functions to be updated. Exact marginalization techniques are tractable in the models proposed in this thesis given their limited complexity.

1.3.2 HMMs

Here we describe the Hidden Markov Model (HMM) [Rabiner, 1989] as a simple example of generative model for time series. Let $(v_1, \ldots, v_m)$ be a sequence of states of an observed random variable $\mathbf{v}$. Furthermore, let $(h_1, \ldots, h_m)$ be the corresponding sequence of states for a discrete hidden variable $\mathbf{h}$ synchronized with $\mathbf{o}$. The joint probability of the sequences of observed and hidden states estimated by an HMM is given by

$$p_{\text{HMM}}(o_1, \ldots, o_m, h_1, \ldots, h_m) = p_\pi(h_1) p_o(v_1|h_1) \prod_{t=2}^{m} p_{\bar{o}}(h_t|h_{t-1}) p_o(v_t|h_t) \ , \tag{1.2}$$

where the $p_{\bar{o}}(.|.)$ terms are called transition probabilities, the $p_o(.|.)$ terms are called emission probabilities, and the $p_\pi(.)$ is the initial probability of the first state of the hidden variable. This model is presented in Figure 1.2, following standard graphical model formalism. Each node is associated to a random variable and arrows denote conditional dependencies. When observed data is discrete, the probability distributions p_π, $p_{\bar{o}}$, and p_o are usually multinomials, whose parameters can be learned efficiently by the EM algorithm

The Gaussian Mixture Model (GMM) is very similar to the HMM. It is commonly used when observing continuous data [Reynolds et al., 2000]. The only difference is that p_o is chosen to be a Gaussian distribution in this case,

to allow continuous observations.

1.4 Motivation

Given infinite amount of data, one could learn the conditional distribution of any random variable in a model given any other random variable. This would lead to a model containing a quadratic number of parameters with respect to the number of random variables in the model. In practical settings, this would lead to models with very high number of parameters, i.e. models with very high *capacity*. Unless provided with extremely high number of examples, such models would inevitably *overfit* data [Vapnik, 1998]. In other words, a model with too many parameters would learn the training set by heart and would fail to *generalize* to unseen data.

Three approaches can be taken to overcome this problem:

1. Build or collect more data;
2. Design better representations for data;
3. Design better algorithms given *a priori* knowledge of the structure of data.

Data

The first approach (i.e. building or collecting more data) may seem the easiest to follow at first glance when designing music models. Millions of audio files are available on the internet. These files usually contains useful meta-data such as artist name, album name, or musical genre. However, these files are usually not labeled consistently, which makes them difficult to use directly to train algorithms for genre classification or information retrieval. Even worse, audio files very rarely contain meta-data directly related to the psychoacoustical characteristics of the music they contain. For instance, very few audio files contain rhythmic, melodic, or harmonic information. As was mentioned in Section 1.2.5, state-of-the-art algorithms are not reliable to extract such information from raw audio data.

The models presented in this thesis are concerned with symbolic music data. Hence, we are limited to existing symbolic databases. The few existing MIDI databases available today are severely limited in size. Moreover, they comprise only specific musical genres. One can not expect to design completely general models of music while learning only from these databases. Nevertheless, there is a huge commercial interest towards modeling pop and jazz music today. This

is due to the dramatic impact that digital portable music players had on the listening habits of a constantly growing number of people around the world. Harmonic and melodic structures in pop music and jazz themes are usually very simple and follow very strong regularities [Sher, 1988].

A dataset of jazz standards (described in Section 2.2.2) was recorded by the author. This dataset is representative of the complexity of common jazz and pop music. It is used in the experiments reported in this thesis along with other public datasets.

Representations

One of the key contributions of this thesis is the design of representations that exhibits important statistical properties of music data. Using appropriate representations is a good way of including domain knowledge in statistical models, and should lead to better generalization.

In Chapter 2, a distributed representation for chords is introduced. This representation is designed such that Euclidean distances correspond to psychoacoustical similarities. In Chapter 3, various chord representations are compared in terms of melodic prediction accuracy. This work is currently under revision in Paiement et al. [2008a]. In Chapter 4, a compressed representation for melodies is introduced for harmonization purposes, along with a corresponding representation for chords. In the following chapters, simple representations for rhythms and melodies are also described as inputs to polyphonic music models. Finally, we describe discrete representations of groups of three melodic notes based on musicological theory in Chapter 6. We show that such representations can be modeled more easily than actual sequences of notes.

Learning Algorithms

Having access to sufficiently large datasets and to reliable representations of these observations is the basis of any machine learning system. However, most of this thesis is concerned with the most important part of statistical analysis, which is the design and evaluation of algorithms themselves.

In Chapter 2, we introduce two distinct graphical model topologies aimed to capture global dependencies in chord progressions. We show empirically that such dependencies are present in music data, and that the proposed models are able to discover such dependencies more reliably than with a simpler HMM. This work is already published in Paiement et al. [2005b] and Paiement et al. [2005a].

In Chapter 4, we design a somewhat complex graphical model of the probabilistic relationships between melodies and chord progressions. This model can be sampled given any melody to generate an accompaniment in the same genre as a training corpus. This work is already published in Paiement et al. [2006].

Then, in Chapter 5, a generative model of the distance patterns between subsequences is proposed. Instead of modeling the rhythms themselves, we propose to model the distributions of the pairwise distances between rhythms. Such a model is then used to put constraints on a local model of the sequences. We show empirically that using such constraints on distances significantly increase out-of-sample prediction accuracy. This work is published in Paiement et al. [2008c].

As a final step, we propose in Chapter 6 a generative model for melodies, given chord progressions and rhythms. This model is based on constraints imposed by a feature representation of groups of three notes. We show that using these constraints leads to much better prediction accuracies than using a simpler Input/Output HMM. Moreover, sampling the proposed model generates melodies that are much more realistic than sampling a local model. This work is currently under revision in Paiement et al. [2008b]. Finally, we describe how to build a full model of polyphonic music by combining all the components presented in the various chapters of this thesis.

2 Chord Progressions

In this chapter, we present two graphical models that capture the chord structures in a given musical style using as evidence a limited amount of symbolic MIDI data. As stated in Section 1.3.1, one advantage of generative models is their flexibility, suggesting that our models could be used either as analytical or generative tool to model chord progressions. Moreover, model like ours can be integrated into more complex probabilistic transcription model, genre classifier, or automatic composition (c.f. Section 1.2.5 for thorough references.)

Chord progressions constitute a fixed, non-dynamic structure in time. As stated in Section 1.2, there is a strong link between chord structure and the much more complex overall musical structure. This motivates our attempt to model chord sequencing directly. The space of sensible chord progressions is much more constrained than the space of sensible melodies, suggesting that a low-capacity model of chord progressions could form an important part of a system that analyzes or generates melodies (see for instance Chapter 6). As an example, consider blues music. Most blues compositions are variations of a basic *same* 12 bar chord progression. Identification of that chord progression in a sequence would greatly contribute to genre recognition. Note that in this thesis, chord progressions are considered relative to the key of each song. Thus, transposition of a whole piece has no effect on our analysis.

In Section 2.1, we briefly present previous work [Raphael and Stoddard, 2004] about probabilistic modeling of chord progressions. In Section 2.2, a distributed representation for chords is designed such that Euclidean distances roughly correspond to psychoacoustic similarities. Graphical models observing chord progressions are then compared in terms of conditional out-of-sample likelihood. Then, in Section 2.3, estimated probabilities of chord substitutions are derived from the same distributed representation. These probabilities are

used to introduce smoothing in graphical models observing chord progressions. Parameters in the graphical models are learnt with the EM algorithm and the classical Junction Tree algorithm is used for inference. Again, various model architectures are compared in terms of conditional out-of-sample likelihood. Both perceptual and statistical evidence show that binary trees related to meter are well suited to capture chord dependencies.

2.1 Previous Work on Chord Progressions Models

Few previous work has been done so far towards probabilistic modeling of chord progressions. In Allan and Williams [2004], a model of chord progressions *given* melodies is proposed. We present this model in Section 4.1. In this section, we briefly describe the graphical model proposed by Raphael and Stoddard [2004] for labeling MIDI data with traditional Western chord symbols.

The analysis is performed on fixed musical periods q, say a measure ($q = 1$), or half measure ($q = 1/2$). The pitches are partitioned into sequences of subsets $y_1 \ldots, y_N$ where $y_n = \{y_n^1, \ldots, y_n^{K_n}\}$ is the collection of pitch-classes whose onset times, in measures, lie in the interval $[nq, (n+1)q]$.

The goal is to associate a key and chord describing the harmonic function to each period y_n. Each y_n will be labeled with an element of

$$L = T \times M \times C = \{0, \ldots, 11\} \times \{\text{major}, \text{minor}\} \times \{\text{I}, \text{II}, \ldots, \text{VII}\}$$

where T, M, C stands for tonic, mode, and chord. For instance, $(t, m, c) = (3, \text{major}, \text{II})$ would represent the triad in the key of $2 = d$ major build on the II = 2nd scale degree which contains pitch-classes e,g,b.

Let $X_1 \ldots, X_N$ be the sequence of harmonic labels $X_n \in L$. Raphael and Stoddard [2004] model this sequence probabilistically as a homogeneous Markov chain

$$p(x_{n+1}|x_1, \ldots, x_n) = p(x_{n+1}|x_n) \ . \tag{2.1}$$

The second assumption is that each data vector y_n only depend on the current label:

$$p(y_n|x_1, \ldots, x_n, y_1, \ldots, y_{n-1}) = p(y_n|x_n) \ . \tag{2.2}$$

The joint model of observed pitches and inferred labels is thus a standard HMM as described in Section 1.3.2. Equation (2.1) is the transition probability distribution while Equation (2.2) is the emission probability distribution. Efficient parameterization are proposed in Raphael and Stoddard [2004] to model these distributions.

The Markovian assumption in Equation (2.1) seems sufficient to infer chord symbols, but we show in Section 2.2.2 that longer term dependencies are necessary to model chord progressions by themselves in a generative context, without regard to any form of analysis.

2.2 A Distributed Representation for Chords

The research reported in this Section was already published in Paiement et al. [2005b].

As pointed out in Section 1.4, the generalization performance of a generative model depends strongly on the chosen representation for observed chords. A good representation encapsulates some of the psychoacoustic similarities between chords. One possibility we chose not to consider was to represent directly some attributes of Western chord notation such as "minor", "major", "diminished", etc. Though inferring these chord qualities could have aided in building a similarity measure between chords, we found it more convenient to start by building a more general representation directly tied to the acoustic properties of chords. Another possibility for describing chord similarities is set-class theory, a method that has been compared to perceived closeness [Kuusi, 2001] with some success. In this Section, we consider a simpler approach where each group of observed notes forming a chord are seen as a single timbre [Vassilakis, 1999] and we design a continuous distributed representation where close chords with respect to Euclidean distance tend to be similar to listeners.

The frequency content of an idealized musical note i is composed of a fundamental frequency $f_{0,i}$ and integer multiples of that frequency. The amplitude of the h-th harmonic $f_{h,i} = hf_{1,i}$ of note i can be modeled with geometric decaying ρ^h, with $0 < \rho < 1$ [Valimaki et al., 1996].

Consider the function

$$m(f) = 12(\log_2(f) - \log_2(8.1758))$$

that maps frequency f to MIDI note $m(f)$. Let $\mathbb{X} = \{\mathcal{X}_1 \dots \mathcal{X}_s\}$ be the set of the s chords present in a given corpus of chord progressions. Then, for a given chord $\mathcal{X}_j = \{i_1, \dots, i_{t_j}\}$ with t_j the number of notes in chord $\mathcal{X}_j$, we associate to each MIDI note n a perceived loudness

$$l_j(n) = \max_{h \in \mathbb{N}, i \in \mathcal{X}_j} (\{\rho^h | \texttt{round}(m(f_{h,i})) = n\} \cup \{0\}) \tag{2.3}$$

where the function `round` maps a real number to the nearest integer. The max function is used instead of a sum in order to account for the masking effect

[Moore, 1982]. The quantization given by the rounding function corresponds to the fact that most of the tonal music is composed using the *well-tempered tuning*. For instance, the 3rd harmonic $f_{3,i}$ corresponds to a note $i + 7$ which is located one perfect fifth (i.e. 7 semi-tones) over the note i corresponding to the fundamental frequency. Building the whole set of possible notes from that principle leads to a system where flat and sharp notes are not the same, which was found to be impractical by musical instrument designers in the baroque era. Since then, most Western musicians used a compromise called the well-tempered scale, where semi-tones are separated by an equal ratio of frequencies. Hence, the rounding function in Equation (2.3) provides a frequency quantization that corresponds to what an average contemporary music listener experiences on a regular basis.

For each chord $\mathcal{X}_j$, we then have a distributed representation

$$\mathbf{l}_j = \{l_j(n_1), \ldots, l_j(n_d)\}$$

corresponding to the perceived strength of the harmonics related to every note n_k of the well-tempered scale, where we consider the d first notes of this scale to be relevant. For instance, one can set the range of the notes n_1 to n_d to correspond to audible frequencies. Using octave invariance, we can go further and define a chord representation

$$\mathbf{v}_j = \{v_j(0), \ldots, v_j(11)\} \tag{2.4}$$

where

$$v_j(i) = \sum_{n_k : 1 \leq k \leq d,\ (n_k \bmod 12) = i} l(n_k).$$

This representation gives a measure of the relative strength of each pitch classin a given chord. For instance, value $v_j(0)$ is associated with pitch class `c`, value $v_j(1)$ to pitch class `c` sharp, and so on. We see in Figure 2.1 that this representation gives similar results for two different voicings of the C major chord, as defined in Levine [1990].

We have also computed Euclidean distances between chords induced by this representation and found that they roughly correspond to perceptual closeness, as shown in Table 2.1. Each column gives Euclidean distances between the chord in the first row and some other chords that are represented as described here. The trained musician should see that these distances roughly correspond to perceived closeness. For instance, the second column is related to a particular inversion of the C minor chord (`c1d#2a#2d3`). We see that the closest chord in the dataset (`c1a#2d#3g3`) is the second inversion of the same chord, as

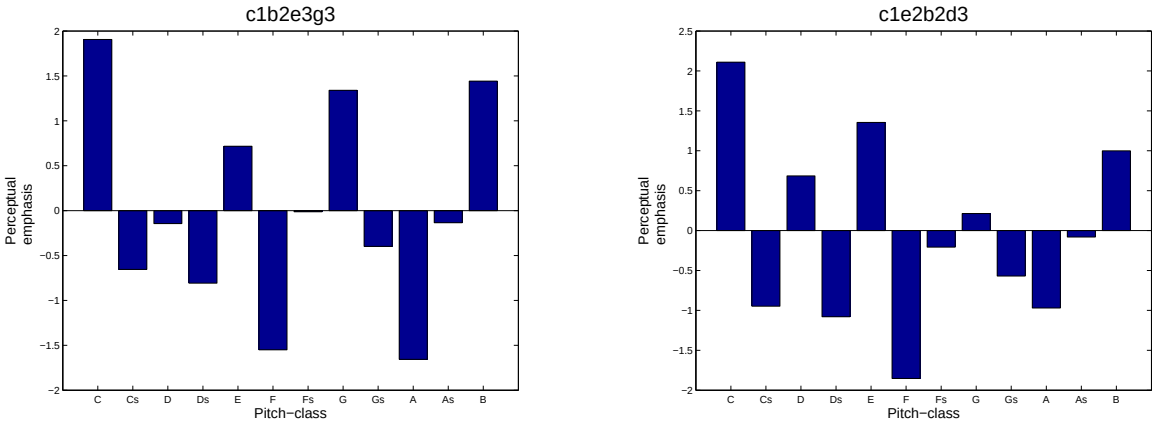

Figure 2.1. Normalized values given by Equation (2.4) for 2 voicings of the C major chord. We see that perceptual emphasis is higher for pitch-classes present in the chord. These two chord representations have similar values for pitch-classes that are not present in either chords, which makes their Euclidean distance small.

described in Levine [1990]. Hence, we raise the note d#3 by one octave and replace the note d3 by g3 (separated by a perfect fourth). These two notes share some harmonics, leading to a close vectorial representation. This distance measure could have considerable interest in a broad range of computational generative models in music as well as for music composition.

2.2.1 Graphical Model

We now propose a graphical model that generates chord sequences using the input representation described in the last section. The main assumption behind the proposed model is that conditional dependencies between chords in a typical chord progression are strongly tied to the metrical structure associated to it.

Table 2.1. Euclidean distances between the chord in the first row and other chords when chord representation is given by Equation (2.4), choosing $\rho = 0.97$.

c1a2e3g3	0.000	c1d#2a#2d3	0.000
c1a2c3e3	1.230	c1a#2d#3g3	1.814
c1a2d3g3	1.436	c1e2a#2d#3	2.725
c1a1d2g2	2.259	c1a#2e3g#3	3.442
c1a#2e3a3	2.491	c1e2a#2d3	3.691
a0c3g3b3	2.920	a#0d#2g#2c3	3.923
c1e2b2d3	3.162	a#0d2g#2c3	4.155
c1g2c3e3	3.398	g#1g2c3d#3	4.363
a0g#2c3e3	3.643	c1e2a#2c#3	4.612
c1f2c3e3	3.914	a#1g#2d3g3	4.820
c1d#2a#2d3	4.295	f1a2d#3g3	5.030
e1e2g2c3	4.548	d1f#2c3f3	5.267
g1a#2f3a3	4.758	a0c3g3b3	5.473
e0g2d3f#3	4.969	g1f2a#2c#3	5.698
f#0e2a2c3	5.181	b0d2a2c3	5.902
g#0g2c3d#3	5.393	e1d3g3b3	6.103
f#1d#2a2c3	5.601	f#1e2a#2d#3	6.329
g0f2b2d#3	5.818	d#1c#2f#2a#2	6.530
g1f2a#2c#3	6.035	g#0b2f3g#3	6.746
g1f2b2d#3	6.242	b0a2d#3g3	6.947

Another important aspect of this model is that it is not restricted to local dependencies, like a simpler Hidden Markov Model (HMM) would be. This choice of structure reflects the fact that a chord progression is seen in this model as a two dimensional architecture. Every chord in a chord progression depends both on its position in the chord structure (global dependencies) and on the surrounding chords (local dependencies.) We show in Section 2.2.2 that considering both aspects leads to better generalization performance as well as better generated results than by only considering local dependencies. Figure 2.2 shows a graphical model that can be used as a generative model for chord progressions in this fashion.

Discrete nodes in levels 1 and 2 are unobserved. The purpose of the nodes in level 1 is to capture global chord dependencies related to the meter. Nodes

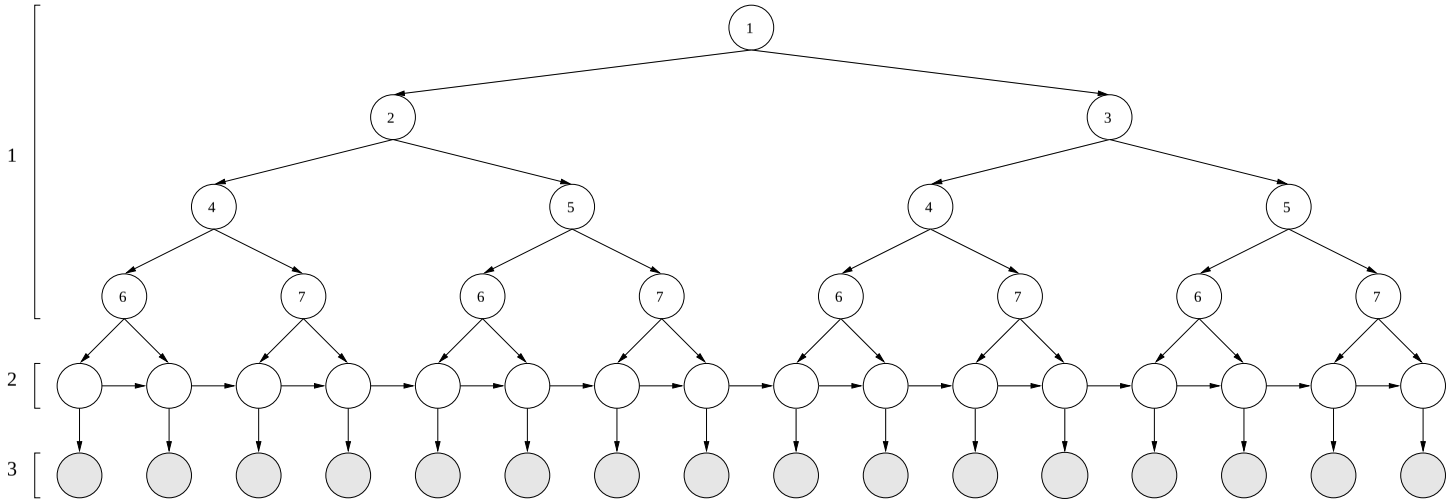

Figure 2.2. A probabilistic graphical model for chord progressions. White nodes correspond to discrete hidden variables while gray nodes correspond to observed multivariate mixtures of Gaussian nodes. Nodes in level 1 directly model the contextual dependencies related to the meter. Nodes in level 2 combine this information with local dependencies in order to model smooth chord progressions. Finally, continuous nodes in level 3 are observing chords embedded in the continuous space defined by Equation (2.4). Numbers in level 1 nodes indicate a particular form of parameter sharing that has proven to be useful for generalization (see Section 2.2.2).

in level 2 are modeling local chord dependencies conditioned by the global dependencies captured in level 1. For instance, the fact that the algorithm is accurately generating proper endings is constrained by the upper tree structure. On the other hand, the smoothness of the voice leadings (e.g. small distances between generated notes in two successive chords) is modeled by the horizontal links in level 2.

The bottom nodes of the model are continuous observations conditioned by discrete hidden variables. Hence, a mixture of Gaussians can be used to model each observation given by the distributed representation described in Section 2.2. Suppose a Gaussian node G has a discrete parent D, then the conditional density $p(G|D)$ is given by

$$p(G|D=i) \sim \mathcal{N}(\mu_i, \sigma_i) \tag{2.5}$$

where $\mathcal{N}(\mu, \sigma)$ is a k-dimensional Gaussian distribution with mean $\mu \in \mathbb{R}^k$ and diagonal covariance matrix $\Sigma \in \mathbb{R}^k \times \mathbb{R}^k$ determined by its diagonal elements $\sigma \in \mathbb{R}^k$.

The inference in the model can be done with the EM algorithm [Dempster et al., 1977] and the marginalization in the graphical model (during the expectation step) can be achieved using the Junction Tree Algorithm [Lauritzen, 1996], as described in Section 1.3.1. Each μ_i and σ_i in Equation (2.5) are learned using the EM algorithm. Exact marginalization techniques are tractable in this model given its limited complexity.

Many variations of this particular model can be possible, some of which are compared in Section 2.2.2. For instance, conditional probability tables can be tied in various ways. Also, more horizontal links in the model can be added to reinforce the dependencies between higher level hidden variables.

Other tree structures may be more suitable for music having different meters (e.g. ternary structures for waltzes). Using a tree structure has the advantage of reducing the complexity of the considered dependencies from the order l to the order $\log l$, where l is the length of a given chord sequence. Although we only consider musical productions with fixed length in this chapter, the proposed models could be easily extended to variable length musical production by adding conditional dependencies arrows between many normalized subtrees.

Tree structures for music have been used previously in Rizo et al. [2006] to find the key of melodic segments. In this work, each measure is represented by a tree where each observed note is represented by a leaf node. The level of each leaf in the tree determines the duration of the associated note. In a 4-beat measure, a leaf on the root level would represent a whole note, a leaf on the

second level would represent a half note, and so on. Pitch-classes are used to label the leaves. Then, these pitch-classes are propagated bottom-up to every parent nodes, using a post-order traversal of the tree. Hence, all the nodes contain the union of the pitch-classes of all their children nodes. Finally, this representation is used to find the key of the corresponding melodic segments with high reliability [Rizo et al., 2006].

In this section, we introduced a similar tree structure related to the meter. However, each node of the current tree is associated to a *hidden random variable*, instead of being labeled with observed pitch-classes. Each of these hidden random variables are modeling global dependencies in chord progressions related to corresponding hierarchical levels of the metrical structure.

2.2.2 Experiments

52 jazz standards excerpts from Sher [1988] were interpreted and recorded by the author in MIDI format on a Yamaha Disklavier piano. See `http://www.idiap.ch/probmusic` for a listing. Standard 4-note jazz piano voicings as described in Levine [1990] were used to convert the chord symbols into musical notes. Thus, the model is considering chord progressions as they might be expressed by a trained jazz musician in a realistic musical context. The complexity of the chord sequences found in the corpus is representative of the complexity of common chord progressions in most pop and jazz music. We chose to record actual voiced chords rather than symbolic chord names (e.g. *Em7*) because the symbolic names are ineffective at capturing the specific voicings made by a trained jazz musician.

Every jazz standard excerpt is 8 bars long, with a 4 beats meter, and with one chord change every 2 beats (yielding observed sequences of length 16.) Longer chords were repeated multiple times (e.g. a 6 beat chord is represented as 3 distinct 2-beat observations.) This simplification has a limited impact on the quality of the model since generating a chord progression is simply a first (but very important) step toward generating complete polyphonic music, where modeling actual event lengths is crucial (c.f. Chapter 5). The jazz standards were carefully chosen to exhibit a 16 bar global structure. We used the last 8 bars of each standards to train the model. Since every standard ends with a cadenza (i.e. a musical ending), the chosen excerpts exhibits strong regularities.

Generalization

The chosen discrete chord sequences were converted into sequences of 12-dimensional continuous vectors as described in Section 2.2. Frequencies ranging from

20Hz to 20kHz (MIDI notes going from the lowest note in the corpus to note number 135) were considered in order to build the representation given by Equation (2.3). A value of ρ of 0.96 was arbitrarily chosen for the experiments. It should be pointed out that since the generative models have been trained in an unsupervised setting, it is irrelevant to compare different chord representations in terms of likelihood or chord prediction error rate. In Chapter 3, we introduce *melodic* prediction error *given* chords as an evaluation criterion for chord representations.

It is however possible to measure how well a given architecture is modeling conditional dependencies between sub-sequences of chords in terms of likelihood. In order to do so, mean negative *conditional* out-of-sample likelihoods of sub-sequences of length 4 on positions 1, 5, 9 and 13 have been computed using double cross-validation.

Cross-validation [Hastie et al., 2001] is a meta learning algorithm usually used when dealing with small datasets. We first divide the dataset into I folds. Then, the main idea is to train the algorithm using $I-1$ folds and to evaluate it on the remaining fold. This process can be repeated over each possible subsets of $I-1$ folds. The average of all the evaluations can be used itself as an evaluation criterion for the learning algorithm.

More formally, assume also that the i-th fold in the cross-validation process contains N_i test sequences. Let $\mathbf{x}^{(n,i)} = (x_1^{(n,i)}, \ldots, x_l^{(n,i)})$ be the n-th sequence of chords of the i-th fold in the cross-validation process. In this case, we have $l = 16$. Also, each $x_t^{(n,i)}$ must be equal to one of the possible values for $\mathbf{v}_j$ in Equation (2.4).

Using cross-validation, our evaluation criterion is given by

$$\frac{1}{I}\sum_{i=1}^{I} -\log\Big(\prod_{n=1}^{N_i}\prod_{t=\{1,5,9,13\}} P(x_t^{(n,i)},\ldots,x_{t+3}^{(n,i)}|x_1^{(n,i)},\ldots,x_{t-1}^{(n,i)},x_{t+4}^{(n,i)},\ldots,x_{16}^{(n,i)})\Big) \tag{2.6}$$

which amounts to

$$\frac{1}{I}\sum_{i=1}^{I}\sum_{n=1}^{N_i}\sum_{t=\{1,5,9,13\}} -\log(P(x_t^{(n,i)},\ldots,x_{t+3}^{(n,i)}|x_1^{(n,i)},\ldots,x_{t-1}^{(n,i)},x_{t+4}^{(n,i)},\ldots,x_{16}^{(n,i)})) \tag{2.7}$$

where $P(.|.)$ is the distribution of a chord sequence given the specified context returned by the graphical model shown in Figure 2.2. We choose this particular measure of generalization to account for the binary metrical structure of the chord progressions in the corpus.

Double cross-validation is a recursive application of cross-validation where both the optimization of the parameters of the model and the evaluation of

Table 2.2. Mean negative conditional out-of-sample log-likelihoods of subsequences of length 4 on positions 1, 5, 9 and 13. These results are computed using double cross-validation in order to optimize the number of possible values for hidden variables. The numbers in parentheses indicate which levels of the tree are tied as described in Section 2.2.2. We see that some combinations of parameter tying in the trees performs better than the standard HMM.

Model (tying)	Negative log-likelihood
Tree (2, 3)	93.8910
Tree (1, 3)	94.0037
Tree (1, 2, 3)	94.9309
Tree (3)	98.2446
HMM	98.2611

the generalization of the model are carried out simultaneously. Standard cross-validation is applied to each subset of $I-1$ folds with each hyper-parameter setting and tested with the best estimated setting on the remaining hold-out fold. The reported conditional likelihoods are the averages of the results of each of the I applications of simple cross-validation during this process. This technique has been used to optimize the number of possible values of hidden variables for various architectures. Results are given in Table 2.2.

We say that two random variables are "tied" when they share the same conditional probability parameters. Different forms of parameter tying for the tree model shown in Figure 2.2 have been tested. All nodes in level 3 share the same parameters for all tested models. Hence, we use only one 12-dimensional mixture of Gaussians (as in Equation (2.5)) independently of time, in order to constrain the capacity of the model. Moreover, a diagonal covariance matrix Σ has been used, thus reducing the number of free parameters to 24 in level 3 (12 for μ and 12 for Σ). Hidden variables in level 1 and 2 can be tied or not. Tying for level 1 is done as illustrated in Figure 2.2 by the numbers inside the nodes.

The fact that the contextual out-of-sample likelihoods presented in Table 2.2 are better for the different trees than for the HMM indicates that time-dependent regularities are present in the data. Sharing parameters in levels 1 or 2 of the tree increases the out-of-sample likelihood. This indicates that regularities are repeated over time in the signal. Further investigations would be necessary in order to assess to what extent chord structures are hier-

archically related to the meter.

On the other hand, the relatively high values obtained in terms of conditional out-of-sample negative log-likelihood indicates that the number of sequences may not be sufficient to efficiently represent the variability of the data. Unfortunately, as pointed out in Section 1.4 reliable chord progressions data is difficult to generate.

Generation

One can sample the proposed model in order to generate genuine chord progressions. Fortunately, Euclidean distances are relevant in the observation space created in Section 2.2. Thus, a simple approach to generate chord progressions is to take the nearest neighbors (in the training set) of the values obtained by sampling the observation nodes.

Chord progressions generated by the models presented in this thesis are available at `http://www.idiap.ch/probmusic`. One can hear that the generated sequences are very similar to standard jazz chord progressions. For instance, the trained musician can observe that the last 8 bars of many sequences are II-V-I chord progression (with lowest notes (roots) `d`, `g` and `c`) [Levine, 1990], which is very common in the training set.

On the other hand, one can also listen to chord progressions generated by the HMM model. While the chords are following each other in a smooth fashion, there is no global relation between chords. For instance, one can see that the lowest note of the last chord is very often not a `c`, which was the case for all the chord sequences in the training set. The fundamental qualitative difference between both methods should be obvious even for the non-musician when listening to the generated chord sequences.

In this Section, we have shown empirically that chord progressions exhibit global dependencies that can be better captured with a tree structure related to the meter than with a simple dynamical HMM that concentrates on local dependencies. The importance of contextual information for modeling chord progressions is even more apparent when one compares sequences of chords sampled from both models. The time-dependent hidden variables enable the tree structure to generate coherent chord progressions both locally and globally.

However, the small difference in terms of conditional out-of-sample likelihood between the tree model and the HMM, and the relatively low optimal capacity for generalization are a good indication that increasing the number of sequences in the dataset would probably be necessary in further developments of probabilistic models for chord progressions. Also, a better evaluation of

such models could be achieved by including them into a supervised task. This problem is explored more thoroughly in Chapter 3, were we introduce melodic prediction as a benchmark task to compare different chord representations.

2.3 A Probabilistic Model of Chord Substitutions

The research reported in this Section was already published in Paiement et al. [2005a].

2.3.1 Probabilities of Substitution

In Section 2.2, we presented a graphical model that directly observes continuous representations of chords. This approach suffers from two drawbacks. First, it is unnatural to compress discrete information in a continuous space; one could easily think of a one-dimensional continuous representation that would overfit any discrete dataset. Second, since the set of likely chords is finite, one would prefer to observe directly discrete variables with a finite number of possible states.

However, there is no direct way to represent Euclidean distances between discrete objects in the graphical model framework. Our proposed solution to this problem is to convert the Euclidean distances between chord representations into probabilities of substitution between chords. Chords can then be represented as individual discrete events. It is possible to convert the Euclidean distances described in Section 2.2 into probabilities of substitution between chords in a given corpus of chord progression. These probabilities can be included directly into a graphical model for chord progressions.

One can define the probability $p_{i,j}$ of substituting chord $\mathcal{X}_i$ for chord $\mathcal{X}_j$ in a chord progression as

$$p_{i,j} = \frac{\phi_{i,j}}{\sum_{1 \leq j \leq s} \phi_{i,j}} \tag{2.8}$$

with

$$\phi_{i,j} = \exp\{-\lambda ||\mathbf{v}_i - \mathbf{v}_j||^2\}$$

with free parameter $0 \leq \lambda < \infty$. Each $\mathbf{v}_i$ is the distributed representation for the i-th chord in the database as computed in Equation (2.4), page 20. The parameters λ and ρ (from Equation (2.3)) can be optimized by validation on any chord progression dataset provided a suitable objective function. With possible values going from 0 to arbitrary high values, the parameter λ allows the substitution probability table to go from the uniform distribution with equal entries everywhere (such that every chord has the same probability of

Table 2.3. Subset of the substitution probability table constructed with Equation (2.8). For each column, the number in the first row corresponds to the probability of playing the associated chord with no substitution. The numbers in the following rows correspond to the probability of playing the associated chord instead of the chord in the first row of the same column.

c1a2e3g3	0.41395	c1d#2a#2d3	0.70621
c1a2c3e3	0.08366	c1a#2d#3g3	0.06677
c1a2d3g3	0.06401	c1e2a#2d#3	0.02044
c1a1d2g2	0.02195	c1a#2e3g#3	0.00805
c1a#2e3a3	0.01623	c1e2a#2d3	0.00582
a0c3g3b3	0.00929	a#0d#2g#2c3	0.00431
c1e2b2d3	0.00679	a#0d2g#2c3	0.00318
c1g2c3e3	0.00500	g#1g2c3d#3	0.00243
a0g#2c3e3	0.00363	c1e2a#2c#3	0.00176
c1f2c3e3	0.00255	a#1g#2d3g3	0.00134
c1d#2a#2d3	0.00156	f1a2d#3g3	0.00102
e1e2g2c3	0.00112	d1f#2c3f3	0.00075
g1a#2f3a3	0.00085	a0c3g3b3	0.00057
e0g2d3f#3	0.00065	g1f2a#2c#3	0.00043
f#0e2a2c3	0.00049	b0d2a2c3	0.00033
g#0g2c3d#3	0.00037	e1d3g3b3	0.00025
f#1d#2a2c3	0.00028	f#1e2a#2d#3	0.00019
g0f2b2d#3	0.00021	d#1c#2f#2a#2	0.00015
g1f2a#2c#3	0.00016	g#0b2f3g#3	0.00011
g1f2b2d#3	0.00012	b0a2d#3g3	0.00008

being played) to the identity matrix (which disallow any chord substitution). Table 2.3 shows substitution probabilities obtained from Equation (2.8) for chords in Table 2.1.

2.3.2 Graphical Model

We now propose a graphical model for chord sequences using the probabilities of substitution between chords described in Equation (2.8). As for the model presented in Section 2.2.1, the main assumption underlying the proposed model

is that conditional dependencies between chords in a typical chord progression are strongly tied to the metrical structure associated with it. Again, another important aspect of this model is that it is not restricted to local dependencies, like a simpler Hidden Markov Model (HMM) [Rabiner, 1989] would be. We show empirically in Section 2.3.3 that considering both aspects leads to better generalization performance as well as better generated results than by only considering local dependencies, following the results reported in Section 2.2.2.

Figure 2.3 shows a graphical model that can be used as a generative model for chord progressions in this fashion. All the random variables in the model are discrete. Nodes in level 1, 2 and 3 are hidden while nodes in level 4 are observed. Every chords are represented as distinct discrete events. Nodes in level 1 directly model the contextual dependencies related to the meter. Nodes in level 2 combine this information with local dependencies in order to model smooth chord progressions. Variables in level 1 and 2 have an arbitrary number of possible states optimized by double cross-validation [Hastie et al., 2001], as explained in Section 2.2.2. The upper tree structure makes it possible for the algorithm to generate proper endings. Smooth voice leadings (e.g. small distances between generated notes in two successive chords) are made possible by the horizontal links in level 2.

The major difference between this graphical model and the one shown in Figure 2.2 is that the last layer of continuous nodes is replaced by two layers of discrete nodes in Figure 2.3. Variables in levels 3 and 4 have a number of possible states equal to the number of chords in the dataset. Hence, each state is associated with a particular chord. The probability table associated with the conditional dependencies going from level 3 to 4 is fixed during learning with the values given by Equation (2.8). Values in level 3 are hidden and represent intuitively "initial" chords that could have been substituted by the actual observed chords in level 4.

The role of the fixed substitution matrix is to raise the probability of unseen events in a way that account for psychoacoustical similarities. Discarding level 4 and directly observing nodes in level 3 would assign extremely low probabilities to unseen chords in the training set. Instead, when observing a given chord on level 4 during learning, the probabilities of *every* chords of the dataset are updated with respect to the probabilities of substitution described in Section 2.3.1.

Again, the marginalization in the graphical model can be achieved using the Junction Tree Algorithm (JTA) [Lauritzen, 1996]. By defining a convenient factorization of all the variables from the one defined by the graph, the JTA

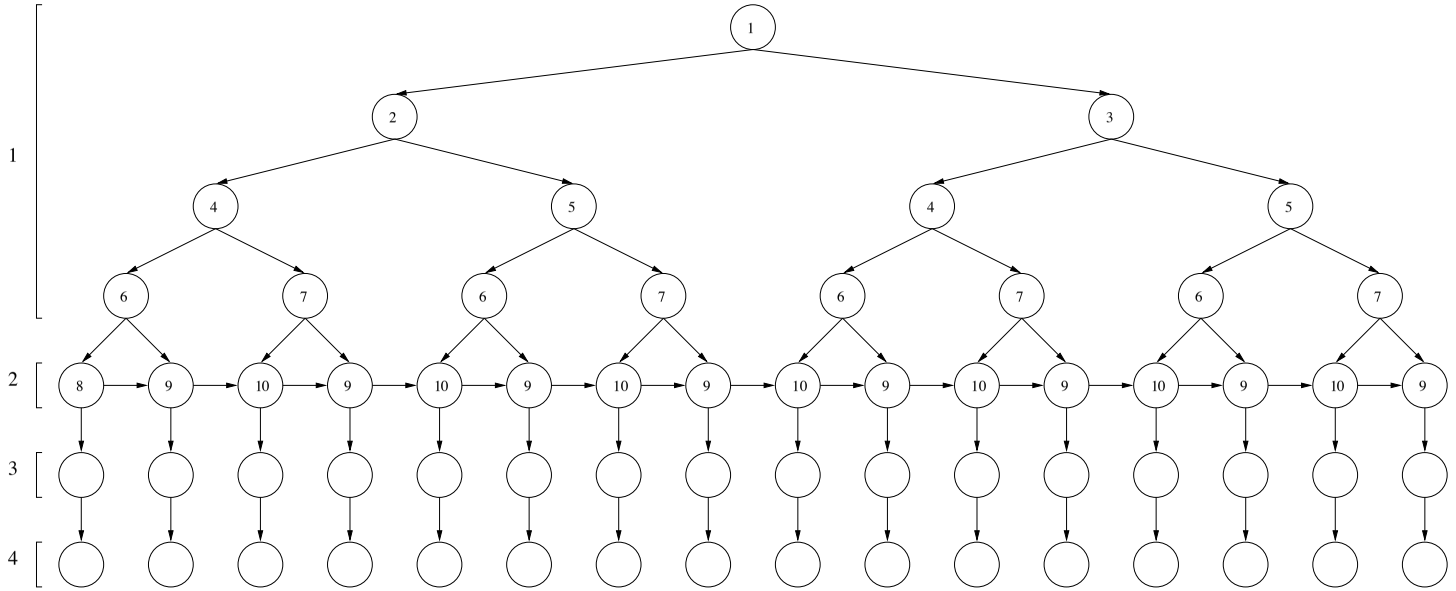

Figure 2.3. A probabilistic graphical model for chord progressions, as described in Section 2.3.2. Numbers in level 1 and 2 nodes indicate a particular form of parameter sharing that has been used in the experiments (see Section 2.3.3).

allows marginalization of small subsets of the variables to be done efficiently. Exact marginalization techniques are tractable in this model given its limited complexity.

Many variations of this particular model are possible, some of which are compared in Section 2.3.3. Conditional probability tables in the proposed model can be tied in various ways. Also, more horizontal links in the model can be added to reinforce the dependencies between higher level hidden variables.

2.3.3 Experiments

The 52 jazz standards excerpts from Sher [1988] described in Section 2.2.2 were used to evaluate the model presented here. However, the length of each sequences was expanded to 32 chords. Hence, every jazz standard excerpt was 16 bars long, with a 4 beat meter, and with one chord change every 2 beats (yielding observed sequences of length 32). We used the last 16 bars of each standard to train the model. Remember that standard 4-note jazz piano voicings as described in Levine [1990] were used to convert the chord symbols into musical notes. Thus, the model is considering chord progressions as they might be expressed by a trained jazz musician in a realistic musical context.

Generalization

The chosen discrete chord sequences were converted into sequences of 12-dimensional continuous vectors as described in Section 2.3.1. Frequencies ranging from 20Hz to 20kHz (MIDI notes going from the lowest note in the corpus to note number 135) were considered in order to build the representation given by Equation (2.3). Again, we want to measure how well a given architecture captures conditional dependencies between sub-sequences. In order to do so, average negative *conditional* out-of-sample likelihoods of sub-sequences of length 8 on positions 1, 9, 17 and 25 have been computed. Equation (2.7) to compute the likelihood of a model now becomes

$$\frac{1}{I}\sum_{i=1}^{I}\sum_{n=1}^{N_i}\sum_{k=\{1,9,17,25\}} -\log(P(x_k^{(n,i)},\ldots,x_{k+7}^{(n,i)}|x_1^{(n,i)},\ldots,x_{k-1}^{(n,i)},x_{k+8}^{(n,i)},\ldots,x_{32}^{(n,i)}))$$

since the sequences of chords have a length of 32. Hence, the likelihood of each subsequence is conditional on the rest of the sequence (taken in the validation set) from which it originates.

Double cross-validation has been used to optimize the number of possible values of hidden variables and the parameters ρ and λ for various architectures. Results are given in Table 2.4.

Table 2.4. Average negative conditional out-of-sample log-likelihoods of subsequences of length 8 on positions 1, 9, 17 and 25, given the rest of the sequences. These results are computed using double cross-validation in order to optimize the number of possible values for hidden variables and the parameters λ and ρ. We see that the trees perform better than the HMM.

Model	(Tying in level 1)	Negative log-likelihood
Tree	No	32.3281
Tree	Yes	32.6364
HMM		33.2527

Two forms of parameter tying for the tree model have been tested. The conditional probability tables in level 1 of Figure 2.3 can be either tied as shown by the numbers inside the nodes in the figure or can be left untied. Tying for level 2 is always done as illustrated in Figure 2.3 by the numbers inside the nodes, to model local dependencies. All nodes in level 3 share the same parameters for all tested models. Also, recall that parameters for the conditional probabilities of variables in level 4 are fixed as described in Section 2.3.2.

As a benchmark, an HMM consisting of levels 2, 3 and 4 of Figure 2.3 has been trained and evaluated on the same dataset. The results presented in Table 2.4 are similar to perplexity or prediction ability. The fact that these contextual out-of-sample likelihoods are better for the trees than for the HMM is an indication that time-dependent regularities are present in the data. This is coherent with the results reported in Section 2.2.2. Further investigations would be necessary in order to measure to what extent chord structures are hierarchically related to the meter.

Conditional likelihood allows to compare quantitatively probabilistic models using the *same* chord representation. However, it is not possible to compare different representations with this experimental setting. As an extreme case, suppose that all different chords are represented by the same symbol. Obviously, this is a bad representation for chords because it destroys all information about the actual chords in the corpus. Despite this fundamental inefficiency, a model dealing with this representation would always have an out-of-sample log-likelihood of 0, thus beating all other possible representations! In other words, the number of possible states of a representation dramatically alters the likelihood of any model observing this representation. This is why negative log-likelihood values are much lower in Table 2.4 (discrete representation

with a finite number of possible values) than in Table 2.2 (continuous representation with an infinite number of potential values). This tendency is even stronger than the fact that sequences used for the experiments reported in Table 2.4 are twice as long as the sequences used for the experiments reported in Table 2.2. Measured over the same representation, negative log-likelihoods would normally tend to be higher for longer sequences. In Chapter 3, we introduce melodic prediction as a supervised task that allows to compare chord representations quantitatively.

Generation

Again, one can sample the joint distribution learned by the proposed model in order to generate novel chord progressions. Chord progressions generated by the models presented in this Section are available at `http://www.idiap.ch/probmusic`. The generated sequences are very similar to the ones generated by the model based on psychoacoustical features presented in Section 2.2.1. When sampling continuous distributions, one has to find the nearest neighbors of each sample in the training set. This approach is arbitrary and lacks statistical justification. In contrast, the generation process described here is more principled: Each sampled chord is simply represented as the discrete state of a random variable.

Both models are able to capture the fact that most endings in the current dataset are II-V-I chord progressions. However, comparing quantitatively generated chord sequences from both models (or any other chord generation model) would require setting up experiments involving human judgment. There is no available mathematical measure of the quality of a chord progression, and it is beyond the scope of this thesis to make an attempt in designing such a measure.

That being said, the fundamental qualitative difference between the chord progressions generated by a "local" method and our proposed model should be obvious even for the non-musician when listening to the generated chord sequences. Chord progressions generated by an HMM follow one another in a smooth fashion. However, there is no global coherence among the chords.

2.4 Conclusion

In this chapter, we have shown empirically that chord progressions exhibit global dependencies that can be better captured with a tree structure related to the meter than with a simple dynamical HMM that concentrates on local

dependencies. The importance of contextual information for modeling chord progressions is even more apparent when one compares sequences of chords sampled from both models. The time-dependent hidden variables enable the tree structure to generate coherent chord progressions both locally and globally.

However, the low difference in terms of conditional out-of-sample likelihood between the tree model and the HMM, and the relatively low number of degrees of freedom for optimal generalization (including the low optimal number of possible states for hidden variables) are a good indication that increasing the number of sequences in the dataset would probably be necessary in further developments of probabilistic models for chord progressions. Also, better evaluation criteria than conditional likelihoods could be used to compare chord probabilistic models. An experimental setting such as in Chapter 3 could be used to compare the choices of parameters to build the distributed representation introduced in this chapter. A supervised task would provide a quantitative evaluation between many distributed representations.

Applications where a chord progression model could be included range from music transcription, music information retrieval, musical genre recognition to music analysis applications, just to name a few.

Chord progressions are regular and simple structures that condition dramatically the actual choice of notes in polyphonic tonal music. Hence, we argue that chord models are crucial in the design of efficient algorithms that deal with such music data. Moreover, generating interesting chord progressions can be considered to be one of the most important aspects in generating realistic polyphonic music. Our proposed chord progression models constitute first steps in that direction.

3 Comparing Chord Representations

As stated in Section 2.3.3, it is impossible to compare quantitatively chord representations in terms of likelihood of an unsupervised model. A data representation has limited value by itself if it is not used as input to an algorithm embedded in a specific application. Ultimately, one has to design a supervised task (e.g. classification) where many representations can be used as inputs to be able to compare representations. Quantitative performance measures available for the particular task can then be considered as a performance measure for the representation itself.

In this chapter, we use melodic prediction as a benchmark to evaluate chord representations. In Section 3.3, we also describe unconditional and conditional prediction error rates as more reliable evaluation criteria than the conditional likelihoods presented in Chapter 2.

The research reported in this chapter is currently under revision for publication in Paiement et al. [2008a].

3.1 Interactions Between Chords and Melodies

Knowing the relations between chords and actual notes would certainly help to discover long-term musical structures in tonal music. Melodies are very often the most salient part of polyphonic music. Hence, we focus here on the probabilistic interactions between chords and melodies.

Very useful applications could follow from an accurate probabilistic model of the relations between chords and melodies. As pointed out in Section 1.2.5, state-of-the-art techniques for transcription from audio to symbolic representation suffer from a lack of precision for most practical applications. An intermediate goal is to try to infer chord symbols from audio data [Bello and Pickens, 2005; Sheh and Ellis, 2003]. This task is simpler than complete transcription of

polyphonic audio signal. Combining reliable chord transcription with a model of the conditional distribution of other musical components (e.g. melodies) given chords could help to improve transcription error rates of existing algorithms. Following the same idea, such models could even be included in genre classifiers or automatic composition systems [Eck and Schmidhuber, 2002] to increase their performance.

What are the usual probabilistic relations between these notational components and the actual sequences of notes in polyphonic music? We explore this issue in this chapter by using melodic prediction given chords as a benchmark task to evaluate the quality of many chord notational components. Likelihoods and conditional and unconditional prediction error rates are used as complementary measures of the quality of each combination of components.

Despite the simple and relatively universal chord building principles, many different notations have been used through music history to represent chord progressions [Dahlhaus, 1990]. We face the same problem in the computer science literature, where each author uses a notation corresponding to his musical background [Allan and Williams, 2004; Paiement et al., 2006; Raphael and Stoddard, 2004; Sheh and Ellis, 2003; Thom, 1995]. All these papers describe arbitrary chord representations embedded in probabilistic models, which are mostly variants of Hidden Markov Models (HMMs). However, to the best of our knowledge, there is no available quantitative comparative study of the effect of the choice of particular chord representations to solve practical applications. Each chord representation carries specific information that could be more adapted to certain tasks (or musical styles) than others. At the same time, all of these notations encapsulate basic information about a chord, such as its root.

3.2 Melodic Prediction Models

In order to assess the effect of using particular chord representations for melodic prediction, we propose two kinds of probabilistic models.

3.2.1 A Local Model

The first proposed strategy is to look at the direct effect of observing particular chord representations without any influence from past observations. In the simple model associated with Figure 3.1, variables in level 1 are associated with chord observations. Their particular form is described in Section 3.2.3. Such variables are observed during training *and* testing. Thus, the dashed horizon-

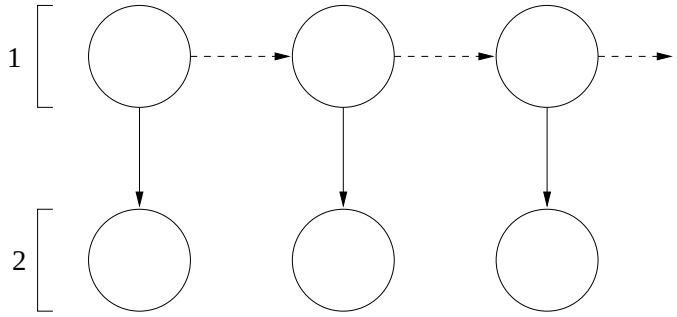

Figure 3.1. A simple probabilistic model where the influence of each chord is direct on melodic observations. When all chord variables in level 1 are observed, the dashed lines are irrelevant, thus making each observation completely local in time. Variables in level 2 correspond to melodic observations.

tal links are not relevant in this context, making each time-step independent of the others. While these variables are discrete, their number of possible values depend on the chosen chord representation, as described in Section 3.2.3. Variables in level 2 correspond to melodic observations. In this chapter, we assume octave invariance for the melodic observations. In other words, every note belonging to the same pitch-class are considered to be the same (e.g. all C notes regardless of octave are associated to the same random variable value).

In level 2, we assign one value to each pitch-class, plus one extra value for silence, leading to a total of 13 possible melodic values. This melodic representation does not account for note similarities. In the proposed models, the probability of the melodic observations given appropriate other random variables is modeled by multinomials. Such a distribution does not embed any notion of similarity between its possible outcomes. However, we chose this melodic representation for its simplicity and also to avoid introducing bias while measuring the quality of the chord representations for melodic prediction.

While this model is overly simplistic for practical purposes, it has the advantage of isolating the direct effect of particular chord representations on the choice of melodic notes. Since all the variables are observed, parameters for this model can be easily computed over a training set by standard maximum likelihood techniques.

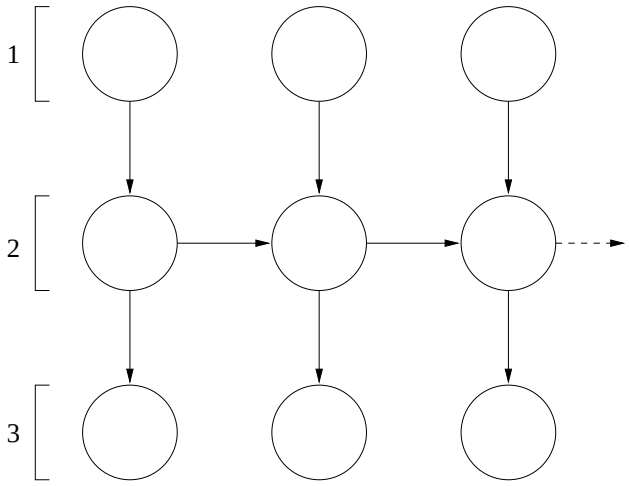

Figure 3.2. Variant of an IOHMM model. The variables in level 1 are observed and correspond to chord observations. Variables in level 2 are hidden, while variables in level 3 correspond to melodic observations.

3.2.2 IOHMM Model

A more realistic model can be designed by adding extra hidden variables in the previous model to consider influences from the past when trying to predict a melody note. The model presented in Figure 3.2 is very similar to an input/output hidden Markov model (IOHMM) [Bengio and Frasconi, 1996].

In our implementation, all the variables in the model are discrete. Variables in level 1 are always observed and again correspond to chord observations. Variables in level 2 are always hidden and are used to introduce dependencies between time frames in the model. Variables in level 3 correspond to melodic observations and have 13 possible values as in the local model presented in Section 3.2.1. There is no link between level 1 and level 3 variables on Figure 3.2, contrary to standard IOHMMs [Bengio and Frasconi, 1996]. The number of possible values is highly variable from one chord representation to another. Considering that, we chose to remove the usual links between inputs and outputs in IOHMMs in order to limit the impact of the particular choice of a chord representation on the capacity of the model. This way, the number of possible values of the chosen chord representation has an impact on the parameterization of the conditional distribution of the hidden variables, but not on the conditional distributions of the predicted melodic variables.

Marginalization must be carried out in the proposed model both for learning (during the expectation step of the EM algorithm) and for evaluation. The

moralization and triangulation steps normally done in the Junction Tree Algorithm are not applicable to the IOHMMs due to their structural simplicity, since no variable has more than one parent and there is no loop in the graph. Exact marginalization is thus tractable in IOHMMs because of their limited complexity.

3.2.3 Chord Representations

The chord representations that we describe in this chapter consider chord symbols as they are represented in musical analysis instead of actual instantiated chords, as in Chapter 2. In other words, we observe chord symbols such as they appear in music sheets [Sher, 1988] instead of observing the notes that would be played by a musician reading these chord symbols. Both kinds of representations may be useful for different applications. However, one should keep in mind that the mapping from instantiated chords to chords symbols is straightforward, if using simple notation. For instance, this is done automatically by most keyboards since the mid 1980's. When using more complex notation, simple probabilistic approaches such as the one described in Section 2.1 [Raphael and Stoddard, 2004] can also be used for harmonic analysis. Hence, a model observing chord symbols instead of actual notes could still be used over traditional MIDI data with minimal preprocessing effort.

Going from chord symbols to instantiated chords is less trivial and depends on the musical style. Levine [1990] proposes simples rules that can be implemented very easily to generate accompaniments from chord symbols. Most keyboards embed accompaniments systems that are automatically able to generate accompaniments in a selected musical genre given appropriate sequences of chord symbols. As we noted in Section 1.2, chords can be seen as latent variables (local in time) that condition the probabilities of choosing particular notes in other music components, such as melodies or accompaniments. The chord symbol "C Maj7" is usually constructed using the pitch classes C, E, G, and B. However, it really defines a conditional probability over *all* pitch classes. For instance, the pitch class D would normally be much more likely over this particular chord than the pitch-class G#. This approach has the advantage of defining directly the chord components as they are conceptualized by a musician. This way, it should be easier in further developments of the proposed models to experiment with more constraints (in the form of conditional independence assumptions between random variables) derived from musical knowledge.

In Chapter 2, we only consider chord progressions by themselves, without

any relationship with other musical components, such as melody or bass lines. In this chapter, we are interested in modeling the relationships between chords and melodies. This is why we restrict ourselves to chord symbols instead of considering actual instantiated chords. Chord symbols have much less possible states than all the potential actual groupings of three or more notes, leading to models with manageable numbers of parameters.

Four chord representations have been used in the experiments described in Section 3.3. First, what we call a `Naive` representation is to consider every chord (including the choice of the root) as a distinct observation. This representation has the disadvantage of excluding any notion of chord similarity. According to this representation, the fact that two chords share some common notes does not have an impact on their associated probabilities. Moreover, this representation leads to a high number of possible values for the associated random variables (e.g. 152 in the current experiments, corresponding to each different chord found in the dataset described in Section 3.3). This can be harmful when learning over small datasets because of the high number of parameters. Despite all these drawbacks, such a representation can be useful if the notions of chord similarities are included in others parts of the models, such as in the conditional probabilities between variables [Paiement et al., 2005a].

Another possible chord representation is to discard any information except the root, yielding random variables with 12 possible values. While having a reasonable number of possible values, such a representation introduces a lot of smoothing in the models. It is shown in Section 4.5 that root progressions already contain valuable contextual information. It is possible to automatically detect the key and the mode (major or minor) of a song [Rizo et al., 2006]. Given that information, the root is very often sufficient to predict the whole structure of the rest of the current chord. For instance, given a song in C major and observing a root C, it is very likely that the complete chord associated to this root is a variant of C major.

We can also restrict ourselves to a subset of all possible chords [Bello and Pickens, 2005; Sheh and Ellis, 2003] by mapping more complex observed chord symbols to a subset of simpler ones. Such a representation is also used in the experiments described in this chapter. We define only whether a chord is minor, major or dominant, leading to chord random variables with only 3 possible values. In this case, if we observe for instance the chord C7#5b9, we map this chord to the value corresponding to a dominant (7) chord. In the remaining of this chapter, this representation is referred to as the `mM7` (**m**inor-**M**ajor-dominant **7**th) representation.

Finally, we can combine the `Root` representation and the `mM7` representation. This leads to discrete random variables with 36 possible values (12 roots times 3 chord qualities).

3.3 Experiments

The same 52 jazz standards that were used for the experiments reported in Chapter 2 were used for the experiments reported in this Section. While the earlier experiments were only involving chord progressions as played by a musician, the current model is considering chord symbols *and* melodies. Hence, the appropriate melodies [Sher, 1988] were interpreted and recorded by the author in MIDI format. Corresponding chord labels were also manually added to this corpus.

Again, the complexity of the chord sequences and melodies found in the corpus is representative of the complexity of common jazz and pop music. The songs were transposed to the key of C. This simplification has no impact on the generality of the experiments since automatic key detection is relatively reliable. Every excerpt was 16 bars long, with four beats per bar, and with one chord change every two beats. Two melodic observations were made for each beat, yielding observed sequences of length 128.

3.3.1 Prediction Error Rate

Cross-validation [Hastie et al., 2001] over a conditional likelihood evaluation criterion is introduced in Equation (2.6), Section 2.2.2. Here, we introduce classification error rate as an alternative evaluation criterion.

Let $x_t^{(n,i)}$ and $c_t^{(n,i)}$ be respectively the melodic observation and the chord observation in sequence n of the test set associated to the i-th fold of the cross-validation process at time t. Assume also that the i-th fold in the cross-validation process contains N_i test sequences, that there is a total of I folds in the cross-validation process and that all the sequences have length T. The rate of error is given by

$$\frac{1}{I}\sum_{i=1}^{I}\frac{1}{N_i}\sum_{n=1}^{N_i}\frac{1}{T}\sum_{t=1}^{T}d_t^{(n,i)} \tag{3.1}$$

where

$$d_t^{(n,i)} = \begin{cases} 1 & \text{if } (\max_{x\in X} p_t^{(n,i)}(x|c_1^{(n,i)},\ldots,c_t^{(n,i)})) \neq x_t^{(n,i)} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

with X being the set of possible melodic observations and $p_t^{(n,i)}(x|c_1^{(n,i)},\ldots,c_t^{(n,i)})$ being the probability of observing melodic value x in sequence n of the test set

associated to the i-th fold of the cross-validation process at time t, estimated by the evaluated model. In words, the out-of-sample error is just the average number of times the algorithm makes a mistake when trying to predict melodic observations over songs *unseen* during training.

It should be pointed out that the classification (or prediction) error is really the criterion we want to minimize when developing models for prediction applications. In such a context, the system *must* make a decision at each time step, which would be the case when using it in almost any realistic context. As an example, a melodic model could be appended to a transcription algorithm. In such a context, the model would have to guess what is the most likely next note, given the previous notes and the audio measurements.

On each iteration of cross-validation, one fold of the dataset is not used for training. This subset of the dataset (referred to as the *test set* in machine learning literature) can be used to evaluate the model. Hence, it is always possible to observe a chord symbol during evaluation (or testing) that has not been observed previously when training the model. Dirichlet priors [Heckerman et al., 1994] have been used on all the chord variables in the algorithms described in this chapter in order to avoid propagating infinite negative log-probabilities in the models in this case. Intuitively, a Dirichlet prior over a multinomial amounts to consider that we have seen *every* possible observations a certain number of times (which may not be an integer) *before* observing the training data.

3.3.2 Local Model

Parameters are tied over time in the `Local` model presented in Figure 3.1. In other words, the arrows between level 1 and level 2 always correspond to the same probability table in one fold of the cross-validation.

Out-of-sample classification errors for the local model described in Section 3.2.1 and for each of the chord representations introduced in Section 3.2.3 are presented in Table 3.1. 5-fold cross-validation was used in the experiments reported in this table. Note that we condition over all previous chords in Equation (3.2), while a melodic prediction in the `Local` model only depends on the current chord observation.

As a benchmark, we also introduce the `Freq` model in Table 3.1, which is simply an algorithm that always selects the most frequent melodic observation in the training set. It corresponds to removing level 1 random variables in Figure 3.1, just leaving independent observations.

With the `Local` model, it is impossible to optimize capacity because no

Table 3.1. Out-of-sample classification error and average negative log-likelihood for local models

Model	**% of Error**	**Neg. Log-Likelihood**
Naive	**77.39**	357.48
Roots + mM7	80.93	316.33
Roots	81.40	**293.95**
mM7	82.41	295.29
Freq	83.34	301.40

parameter is available which relates to the number of degrees of freedom in the model. Thus in the `Local` model, the only way to increase capacity is to use a chord representation with more possible values. Looking at the obtained results, we see that the rate of error increases when using chord representations with *fewer* possible values. It is then possible that all these models somehow underfit, making the `Naive` representation (with 152 possible chord values in the current experiments) the best choice in this context. Not surprisingly, the `Freq` model, which always chooses the most frequent melodic observation, is the worst model in terms of classification error rate. Generalization capabilities of a model can benefit from the smoothing produced by a simplified chord representation, since this simplification is done by clustering perceptual properties of chords (e.g. all chords with the same root are clustered together in the `Root` representation). This is obviously not the case when using the `Freq` model.

Table 3.1 also shows the average negative out-of-sample log-likelihoods obtained with the same models again using cross-validation. Using the same notation as in Equation (3.1) and (3.2), the conditional likelihood criterion introduced in Equation (2.6) now becomes

$$\frac{1}{I}\sum_{i=1}^{I} -\log\Big(\prod_{n=1}^{N_i}\prod_{t=1}^{T} p_t^{(n,i)}(x_t^{(n,i)}|c_1^{(n,i)},\ldots,c_t^{(n,)})\Big) \tag{3.3}$$

which is equal to

$$\frac{1}{I}\sum_{i=1}^{I}\sum_{n=1}^{N_i}\sum_{t=1}^{T} -\log(p_t^{(n,i)}(x_t^{(n,i)}|c_1^{(n,i)},\ldots,c_t^{(n,i)})). \tag{3.4}$$

This performance measure is the one that was optimized over the training set when learning the parameters of the models with the EM algorithm.

As can be observed, the negative log-likelihood results are not coherent

Table 3.2. Out-of-sample classification error and average negative log-likelihood for IOHMM models

Model	% of Error	Neg. Log-Likelihood
Naive	**79.05**	281.84
Roots	82.09	223.67
Roots + mM7	83.17	247.48
mM7	84.71	212.47
HMM	86.56	**196.27**

with the classification error. For instance, the `Freq` model has a lower negative log-likelihood than the `Roots + mM7` and `Naive` models! This result is counter-intuitive since one would expect that adding current chord information would help the model to guess what would be the current melody note.

These discrepancies between classification error rate and negative log-likelihood could be explained by the fact that each of the terms of the sum on the right hand side of Equation (3.3) is not bounded negatively. Suppose that a model fits most of the data quite well but some of the out-of-sample examples have very low probabilities. Then, the terms associated to these examples in Equation (3.3) can take very large negative values that could dominate the average negative log-likelihood for all the examples. On the other hand, the cost of encountering a very unlikely out-of-sample sequence (with respect to the model being evaluated) in Equation (3.1) is only proportional to $\frac{1}{N}$, with N being the total number of examples in the dataset. This observation raises the following question: Is the likelihood of the model over the observed data the best criterion to optimize when what we *really* want to do is to minimize the error of classification? We further discuss this question in Section 4.6.

3.3.3 IOHMM Model

Table 3.2 shows out-of-sample classification error and average negative log-likelihood for the IOHMM model (Section 3.2.2). These results are qualitatively similar to those in Table 3.1 for the `Local` model.

This time, the number of possible values for hidden variables in level 2 of Figure 3.2 was optimized using 5-fold *double* cross-validation, which is a recursive application of cross-validation where both the optimization of the parameters of the model and the evaluation of the generalization performance of the model are carried out simultaneously, as described in Section 2.2.2. Stan-

dard cross-validation was applied to each subset of 4 folds with each hyper-parameter setting and tested with the best set of parameters (on average) on the remaining hold-out fold.

The same parameters are used over time to define the conditional probability distributions. For instance, all the vertical arrows between variables in level 1 and level 2 in Figure 3.2 represent the *same* probability table. The fact that it was possible in this context to optimize the capacity (i.e. the number of parameters) of the models according to the number of possible values for the hidden variables makes the results in Table 3.2 more trustworthy than the ones found in Table 3.1, although they are similar. It should be pointed out that the capacity of the models was optimized with respect to the appropriate error measure. For instance, when reporting results about prediction error rates, capacity is optimized with respect to prediction error rate (while the models are trained by maximizing the likelihood with the EM algorithm).

The HMM model referred to in Table 3.2 is similar to the IOHMM model but removing the chord inputs layer. It can be represented by the model in Figure 3.1 with the horizontal dashed arrows being present. In this case, variables in level 1 are hidden and variables in level 2 are still melodic observations. As expected, the HMM model produces higher out-of-sample classification error than the IOHMM models, which can take advantage of the chord symbols given as inputs.

Interestingly, the `Naive` representation for chords seems to be consistently efficient for melodic prediction. In Table 3.2, the `Naive` representation gives statistically significantly better results than the `Roots` representation with a confidence level of 99%. We used a standard proportion test, assuming a binomial distribution for the errors and using a normal approximation. This is an indication that developing probabilistic models with this representation could be a viable approach, especially if psychoacoustic relations between chords are included in the models via the conditional probabilities related to these chords, as in Section 2.3 [Paiement et al., 2005a]. The representation including only the roots also performs well. Given these results, this representation appears to be a good compromise given its relative simplicity and the fact that it inherently embodies psychoacoustically relevant smoothing. Using basic chord information (`mM7` representation) does not seem to help with respect to unconditional classification error.

Again, average negative log-likelihoods contradict average classification error rates. Even worse, the HMM model performs much better than the IOHMM models in terms of likelihood! This is an indication that such a measure favors

models that are more uniform in essence, thus giving a relatively high probability to unseen sequences. However, such models are weaker when asked to make a decision, meaning that they define distributions with modes less precisely adapted to the training data.

3.3.4 Conditional Classification Error

The goal of the models presented here is to predict the melodies in the dataset. It is out of the scope of this work to evaluate the subjective artistic quality of the predicted melodies. A more interesting measure of melodic prediction is the out-of-sample *conditional* classification error, given by Equation (3.1), but using

$$d_t^{(n,i)} = \begin{cases} 1 & \text{if } \max_{x \in X} p_t^{(n,i)}(x|c_1^{(n,i)}, \ldots, c_t^{(n,i)}, x_1^{(n,i)}, \ldots, x_{t-1}^{(n,i)}) \neq x_t^{(n,i)} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

This measure is very similar to the unconditional error described in Section 3.3.3. However, the models have access to the true previous melodic observations when trying to guess the next one.

The only objective performance measure we can provide about a melodic prediction given a chord progression is to tell if a predicted melody is similar or not to the one provided in the dataset with the same chord progression. However, while the space of plausible melodies is huge, we only have access to a very small number of sequences to measure the performance of the models given a chord progression. Moreover, given a particular sequence of chords, one can imagine that a very high number of melodies would be considered more or less musically similar to the ones in the dataset. Among all these melodies, some of them may not share a single note with the true melody associated with this sequence of chords in the test set. A good melodic prediction model would be likely to generate any one of these melodies, thus producing a very high unconditional error rate.

The conditional error rate alleviates this problem by measuring the prediction performance of a model in regions of the observation space where data is present, leading to a much more reliable performance measure in this context. Moreover, distributions that would generalize well according to such a measure could also be sampled to generate realistic melodies given chord progressions and initial melodic motives. Out-of-sample conditional classification error rates for the IOHMM models presented in Section 3.2.2 are shown in Table 3.3. We do not provide conditional classification error rates for the `Local` model described in Section 3.2.1, because they would be identical to the unconditional

Table 3.3. Out-of-sample conditional classification error rates for IOHMM models

Model	**% of Error**
Roots	**57.41**
Roots + mM7	58.21
mM7	58.32
Naive	69.77
HMM	85.27

classification error rates shown in Table 3.1. When making a prediction, such models are completely unaware of previous observations in time. Also, we do not provide average negative log-likelihoods, since this measure is not well adapted to prediction tasks, as we noted in Section 3.3.

Again, the HMM model produces higher out-of-sample conditional classification error rate than the IOHMM models which benefit from chord symbols given as inputs. In Table 3.3, the conditional error rates are much lower than the unconditional ones for the same models. For each chord representation, the prediction accuracy gained when observing previous melodic notes is much higher than the differences in error rates for each chord representations obtained in Table 3.2. This means that observing previous melodic notes gives more information about the likely choices for the current melody than any chord information.

In Table 3.2, the `Naive` representation was the best one in terms of unconditional prediction error rate. On the other hand, this representation has the highest conditional prediction error rate among all the IOHMMs. When knowing nothing about the previous melodic observations, the model performs better when provided with a more detailed chord representation. However, given previous melodic observations, smoothed chord representations lead to better generalization in terms of prediction error rate. The `Naive` representation overfits the training data because it leads to models with higher capacity. Finally, no representation is statistically significantly better than another with a confidence level of 90% among the three best representations in Table 3.3.

3.4 Conclusion

The main motivation behind this chapter was to better understand the statistical relations between chord representations and the actual choice of notes in polyphonic music. To this end, we compared four chord representations using melodic prediction as a benchmark task.

Surprisingly, the `Naive` representation where each chord is conceived as a discrete observation apparently performs well in terms of unconditional prediction error rates. However, this representation overfits when past melodic observations are used to condition the predictions. In this case, smoothed chord representations seems more appropriate. Given the obtained results, representing chords only by their roots seems to be a good compromise, especially when all the songs to be analyzed are transposed to the same key. While being extremely simple, this representation inherently includes smoothing related to psychoacoustical relations between notes. Moreover, it is shown later in Section 4.3 that root progressions contain valuable non-local information. The `Root+mM7` representation that is used in some important music information retrieval papers [Bello and Pickens, 2005; Raphael and Stoddard, 2004] is not optimal in any of the experiments. However, in practice, the actual choice of a chord representation should always be made considering the application to be developed in mind.

An interesting observation when looking at the results of the experiments done in Section 3.3 is that the behavior of the average out-of-sample likelihood does not follow the trends of the average prediction error rate (conditional or unconditional). On the one hand, the likelihood is a measure of the fit of a whole distribution to a dataset. However, the classification error seems to be a better descriptor of the fit of the *modes* of a distribution. Provided a nearly infinite amount of data, these two measures would lead to the same ranking of the models. Thus, likelihood and prediction error would probably be more comparable when measured with models trained and evaluated with much more data. How much data is needed in this particular framework to obtain such a behavior is still an open question that needs to be addressed.

However, given realistic datasets, optimizing the likelihood of a model with respect to training data may not be the best strategy when one is only interested in the modes of the distribution, which can be the case when doing prediction. An alternative learning strategy would be to maximize the sum of the differences between the log-probabilities of the observed classes and the probabilities of the most probable wrong classes instead of just maximizing the

sum of the log probabilities of the observed classes. This approach is referred to as the minimum classification error (MCE) algorithm [Juang and Katagiri, 1992].

The probabilistic models presented in this chapter were specifically designed to compare chord representations in terms of melodic prediction. Generated melodies obtained when sampling these models conditional to some given chord progression would not be realistic. We show empirically in Chapter 6 that melodies involve long term dependencies that can not be captured by simple stochastic models, such as the IOHMM models. However, the dual problem, which is to generate chord progressions given melodies (called harmonization), is much simpler and we propose a graphical model to achieve this task in Chapter 4, leading to very good practical results.

4 Harmonization

As an application of probabilistic models of interactions between chords and melodies presented in Chapter 3, we propose a model for harmonization (i.e. choosing appropriate chords given melodies). A nice property of graphical models is that it is possible to compute the conditional marginals of any variable given any set of other variables after appropriate training. Thus, very similar models to the ones used for melodic prediction given chords can be used for chord prediction given melodies. However, slight modifications are required both for melodic and chord representations. The research results presented in this chapter are already published in Paiement et al. [2006].

4.1 Previous Work on Harmonization

Allan and Williams [2004] designed a harmonization model for Bach chorales using Hidden Markov Models. Their proposed model deals exclusively with chorales, where a distinct chord occurs at each time step. A chord *must* be formed with 4 notes, including the melody. They associate each melody note to the observed state of an HMM (grey nodes in Figure 1.2). The remaining three notes are associated to the corresponding hidden state (white nodes in the same figure).

Pitches are represented relative to the current melody note. Specific harmonic labels must be added manually to the corpus, which limit severely the applicability of the algorithm to large datasets. First order assumption is made for the transition between chords. In other words, they assume that the probability of the choice of a particular chord at a given time only depends on the previous chord and the current melody note. We show in Section 2.2.2 that this assumption is too strong: the choice of a chord at a given time depends on longer term dependencies. For instance, the last chord of a song is almost

always the tonic chord. There is no direct way for a local model as an HMM to capture that simple property of chord progressions.

The first step for harmonization is done by using a standard Viterbi algorithm. Given visible states (i.e. the melodic notes on each time-step), the algorithm finds the most likely sequence of hidden chord states.

Chords are defined simply as a specific choice of 4 notes. Since chords occur at each time steps, specific notes have to be chosen at each time steps. This approach is quite unrealistic in general musical settings. The authors overcome this problem by introducing an ornamentation mechanism specific to Bach chorales. Each time step is divided into four parts, where the notes chosen by the already described "chord" HMM can be modified by an "ornamentation" HMM. The visible states of this second HMM observe each intervals between notes already generated by the first HMM. Then, the hidden states correspond to each additional notes that can be added or not on each 4 subdivisions of each time-step. Another Viterbi algorithm can be used to generate a sequence of hidden states given chords found by the "chord" HMM.

While generating excellent musical results on Bach chorales, this model has to be provided polyphonic music with specific 4 voice structure as input, restricting its applicability to very particular settings. Our work in this chapter goes a step further by modeling chord progressions given melodies in a simpler and more general way, through the use of a carefully designed chord representation.

4.2 Melodic Representation

Melodic events tend to happen much more often than chord events. While many note beginnings and endings can happen within a single beat, only one chord change usually happens every two, four, or even eight beats, closely following the metric structure.

In Section 3.3, where we explored melody prediction, chord symbols were simply repeated for each of the time-steps they spanned. In the case of chord prediction, however, we must nearly always deal with the fact that many notes in the melody map onto the same chord. To address this, we introduce a melodic representation that allows us to merge many melodic notes into one random variable. A simple way to achieve this is to represent melodies with a 12-dimensional continuous vector representing the relative importance of each pitch class over a given period of time t. We first observe that the lengths of the notes comprising a melody have an impact on their perceptual emphasis. Usually, the meter of a piece can be subdivided into small time-steps such that

the beginning of any note in the whole piece will approximately occur on one of these time-steps. For instance, let t be the time required to play a whole bar. Given that a 4-beat piece (where each beat is a quarter note in length) contains only eighth notes or longer notes, we could divide every bar into 8 time-steps with length $t/8$ and every notes of the piece would occur approximately on the onset of one of these time-steps occurring at times $0, t/8, 2t/8, \ldots, 7t/8$ (i.e. we do *quantization*). Then we can assign to each pitch-class a perceptual weight equal to the total number of such time-steps it covers during time t.

However, as said previously, it turns out that the perceptual emphasis of a melody note depends also on its position related to the meter of the piece. We illustrate in Table 4.1 a way of constructing a weight vector assessing the relative importance of each time-step in a 4-beat measure divided into 12 time-steps with "swinging" eight notes, relying on how meter interacts with musical rhythm [Cooper and Meyer, 1960]. To play "swinging" eighth notes, one just delays the beginning of each note that would normally occur on half the length of a beat (following standard notation) to approximately two-thirds of the length of the same beat. The goal of this weight vector is to quantify the perceptual emphasis of each possible positions in a measure. To construct this representation, we point out iteratively one or more positions that have less perceptual emphasis than the previous added ones. Each step is represented by a row in the table. We sum the number of time each position appears in the process in the resulting vector on the last row. Hence, this vector accounts for the perceptual emphasis that we apply to each time-step in the measure.

Suppose that we observe the note D on the first beat, the note G on the second beat, and the note C on the two last beats. We would have a melodic representation equal to $(13, 0, 8, 0, 0, 0, 0, 6, 0, 0, 0, 0)$ for the whole measure, given the weights in Table 4.1 (if the first index correspond to pitch-class C, and so on). For instance, we sum the numbers in the first three positions $(5+1+2 = 8)$ to obtain the vectorial representation for the note D. Although this method is based loosely on widely accepted musicological concepts, more research would be needed to assess its statistical reliability and to find optimal weighting factors.

4.3 Modeling Root Note Progressions

As it was shown in Table 3.3, one of the most important notes in a chord with regard to its interaction with the melody is the root. For example, bass players play the root note of the current chord very often when accompanying other musicians in a jazz or pop context. As a first step towards complete

Table 4.1. A way to construct a vector assessing the relative importance of each time-step in a 4-beat measure divided into 12 time-steps

Beat	1	.	.	2	.	.	3	.	.	4	.	.
	.											
	.						.					
	.			.			.			.		
	.		.	.		.	.		.	.		.
	.	.	.	.	.	.	.	.	.	.	.	.
	5	1	2	3	1	2	4	1	2	3	1	2

On each row, we add positions that have less perceptual importance than the previous added ones, ending with a weight vector covering all the possible time-steps.

harmonization, Figure 4.1 shows a model that learns interactions between root notes (or chord names) and the melody.

Discrete nodes in levels 1 and 2 are not observed. The upper tree structure is the most striking difference between this graphical model and the ones introduced in Section 3.2 for melodic prediction. The purpose of the nodes in level 1 is to capture global chord dependencies related to the meter [Cooper and Meyer, 1960; Paiement et al., 2005b], like in the chord models presented in Chapter 2. Since the goal is eventually to sample chord progressions from this model given melodies, we wanted to enforce global dependencies with such a structure. For instance, the fact that the algorithm is accurately generating proper endings is constrained by the upper tree structure. A quantitative comparison between this tree structure and its local counterpart is presented in Section 4.5. Nodes in level 2 are modeling local chord dependencies conditionally to the more global dependencies captured in level 1.

Such a model is able to predict sequences of root notes given a melody, which is a non-trivial task even for humans. Nodes in level 2 are tied according to the numbers shown inside the vertices. Probabilities of transition between levels 3 and 4 are fixed with probabilities of substitution related to psychoacoustic similarities between notes [Paiement et al., 2005a]. These random variables have 12 possible values corresponding to each possible root note. We model the probability of substituting one root note for another, as in Section 3.2.3. Nodes in level 3 are hidden while nodes in level 4 are observed. Discarding

level 4 and directly observing nodes in level 3 would assign extremely low probabilities to unseen root notes in the training set. Instead, when observing a given chord on level 4 during learning, the probabilities of *every* root note is updated with respect to the fixed probabilities of substitution. Nodes in level 5 are continuous 12-dimensional Gaussian distributions that are also observed during training where we model each melodic observation using the technique presented in Section 4.2.

In order to evaluate the model presented in Figure 4.1, the database described in Section 3.3 has been used. However, this time, each beat was divided into 6 time-steps (leading to 24 time-steps per 4-beats bars) in order to fit each melody note to an onset. The melodic representation presented in Section 4.2 was used over a time span t of 2 beats corresponding to the chords lengths.

The proposed tree model was compared to a local model (built by removing nodes in level 1) in terms of prediction ability *given* the melody. In order to do so, average negative conditional out-of-sample likelihoods of sub-sequences of length 4 on positions 1, 5, 9 and 13 have been computed. For each sequence of chords $\mathbf{x} = \{x_1, \dots x_{16}\}$ in the appropriate validation set, we average the values

$$-\log P(x_i, \dots, x_{i+3} | x_1, \dots, x_{i-1}, x_{i+4}, \dots, x_{16}). \tag{4.1}$$

with $i \in \{1, 5, 9, 13\}$, as in Equation (2.6). Again, the likelihood of each subsequence is conditional on the rest of the sequence taken in the validation set and the corresponding melody.

As mentioned in Section 3.3.3, likelihood is probably not the most informative measure to assess the quality of models designed for prediction. The goal in this case was to make decisions (predicting the most probable melodic observation) based on the modes of the distribution at each time frames. On the other hand, an interesting use of a harmonization algorithm would be to sample repeatedly the estimated distribution to obtain many possible harmonizations of the same melody. Conditional likelihood of the out-of-sample observations is then in this case a more appropriate performance measure than the prediction error rate.

Using double cross-validation, we let the number of possible values for random variables in levels 1 and 2 go independently from 2 to 15. This technique has been used to optimize the number of possible values of hidden variables and results are given in Table 4.2 in terms of average conditional negative out-of-sample log-likelihoods of sub-sequences. We chose this particular structure for conditioning the observations in order to account for the binary metrical structure of chord progressions, which is not present in natural language processing,

Table 4.2. Average conditional negative out-of-sample log-likelihoods of subsequences of root notes of length 4 on positions 1, 5, 9 and 13 *given* melodies.

Model	Negative log-likelihood
Tree	6.6707
Local	8.4587

These results are computed using double cross-validation in order to optimize the number of possible values for hidden variables. The results are better (lower negative likelihood) for the tree model than for the local model.

for instance.

The fact that results are better for the tree model than for the local model tells us that non-local dependencies are present in root note progressions [Paiement et al., 2005b]. Generated root note sequences given out-of-sample melodies are presented in Section 4.5 together with generated chord structures.

4.4 Decomposing the `Naive` Representation

Before describing a complete model for harmonization, we introduce in this section a decomposition of the `Naive` chord representation introduced in Section 3.2.3 that allows us to model dependencies between each chord component and the proper pitch-class components in the melodic representation presented in Section 4.2.

Each chord is represented by a root note component (which can have 12 possible values given by the pitch-class of the root note of the chord) and 6 structural components detailed in Table 4.3. It is out of the scope of this thesis to describe jazz chord notation in detail [Levine, 1990], but we note that there exists a one-to-one mapping between the chord representation introduced in Table 4.3 and the values of the `Naive` representation, in which each discrete value corresponds to distinct observed chord symbols [Sher, 1988] (see example below). However, these two representations are not equivalent. The decomposition allows us to model directly the impact of each chord components on other random variables through appropriate conditional probability parameterizations, which is not the case with the simpler `Naive` representation.

For the trained musicians, we show in Table 4.4 the mappings of some chord symbols to structural vectors according to this representation. For instance,

Table 4.3. Interpretation of the possible values of the structural random variables.

	Values			
Component	1	2	3	4
3rd	M	m	sus	-
5th	P	b	#	-
7th	no	M	m	M6
9th	no	M	b	#
11th	no	#	P	-
13th	no	M	-	-

Possible values for our structural decomposition of chords. For instance, the variable associated to the 5th of the chord can have 3 possible values. Value 1 corresponds to the perfect fifth (P), value 2 to the diminished fifth (b) and value 3 to the augmented fifth (#). While explaining harmony theory is out of the scope of this thesis, we provide this table to detail the proposed chord structural decomposition to trained musicians.

an observed chord may have a root C, a major third (E), a perfect fifth (G), and no other note. This would correspond to one single arbitrary value of the `Naive` representation. The same chord would be represented by 7 values in the decomposed representation, one value for the root and 6 structural components, namely $(1, 1, 1, 1, 1, 1)$ (i.e. a major third, a perfect fifth, no seventh, no ninth, no eleventh, and no thirteenth), referring to the mapping presented in Table 4.3.

The fact that each structural random variable has a limited number of possible values will produce a model that is computationally tractable. While such a chord decomposition may not look general for a non-musician, we believe that it is applicable to most tonal music by introducing proper chord symbol mappings. Moreover, as pointed out previously, it allows us to directly model the dependencies between chord components and melodic components.

4.5 Chord Model given Root Note Progression and Melody

Figure 4.2 shows a probabilistic model designed to predict chord progressions *given* root note progressions and melodies. The nodes in level 1 are discrete hidden nodes as in the root notes progressions model. The gray boxes are subgraphs that are detailed in Figure 4.3.

Table 4.4. Mappings from some chord symbols to structural vectors according to notation described in Table 4.3

Symbol	3rd	5th	7th	9th	11th	13th
6	1	1	4	1	1	1
M7	1	1	2	1	1	1
m7b5	2	2	3	1	1	1
7b9	1	1	3	3	1	1
m7	2	1	3	1	1	1
7	1	1	3	1	1	1
9#11	1	1	3	2	2	1
m9	2	1	3	2	1	1
13	1	1	3	2	1	2
m6	2	1	4	1	1	1
9	1	1	3	2	1	1
dim7	2	2	4	1	1	1
m	2	1	1	1	1	1
7#5	1	3	3	1	1	1
9#5	1	3	3	2	1	1

Mappings for structural decomposition of chords. For instance, the chord with symbol 7#5 has a major third (M), an augmented fifth (#), a minor seventh (m), no ninth, no eleventh and no thirteenth.

The H node is a discrete hidden node modeling local dependencies and corresponding to the nodes on level 2 in Figure 4.2. The R node corresponds to the current root note. This node can have 12 different values corresponding to the pitch class of the root note and it is always observed. Nodes labeled from 3rd to 13th correspond to the structural chord components presented in Section 4.4. Node B is another structural component corresponding to the bass notation (e.g. G7/D is a G seventh chord with a D on the bass). This random variable can have 12 possible values defining the bass note of the chord. All the structural components are observed during training to learn their interaction with root note progressions and melodies. These are the random variables we try to predict when using the model on out-of-sample data. The nodes on the last row labeled from 0 to 11 correspond to the melodic representation introduced in Section 4.2.

Table 4.5. Average negative conditional out-of-sample log-likelihoods of subsequences of chord structures

Model	Negative log-likelihood
Tree	9.9197
Local	9.5889

Chord structures have length 4 and are taken on positions 1, 5, 9 and 13. Likelihoods are conditional on the rest of the current sequence, the complete root note progressions and complete melodies. Results are computed using double cross-validation.

It should be noted that the melodic components are observed *relative* to the current root note. In Section 4.3, the model is observing melodies with absolute pitch, such that component 0 is associated to note C, component 1 to note C#, and so on. On the other hand, in the present model component 0 is associated to the root note defined by node R. For instance, if the current root note is G, component 0 will be associated to G, component 1 to G#, component 2 to A, and so on. This approach is necessary to correctly link the structural components to the proper melodic components as shown by the arrows between the two last rows of nodes on Figure 4.3.

It is possible to evaluate the prediction ability of the model for chord structures. We present in Table 4.5 the average negative conditional out-of-sample log-likelihoods of chord structures of length 4 on positions 1, 5, 9 and 13, given the rest of the sequence, the complete root note progressions and the melodies for the tree model and the local model built by removing the nodes in level 1 in Figure 4.2.

Again, we used double cross-validation in order to optimize the number of hidden variables in the models. We observe that the local model gives better results than the tree model in this case. This can be explained by the fact that the root note progressions are given in these experiments. This would mean that most of the contextual information would be contained in the root note progression, which makes sense intuitively. Further statistical experiments could be done to investigate this behavior. Table 4.6 shows three different harmonizations of the last 8 measures of the jazz standard *Blame It On My Youth* [Sher, 1988] generated by the proposed model.

When observing the predicted structures given the original root note pro-

Table 4.6. Three different harmonizations of the last 8 measures of the jazz standard *Blame It On My Youth*

OC (1-8)	AbM7	Bb7	Gm7	Cm7	Fm7	Fm7/Eb	Db9#11	C7
OR	AbM7	Bb7	Gm7	C7	Fm7	Fm7	Db7	Cm7
NH	C7	C7	Gm7	Gm7	Fm7	Fm7	Bb7	Bb7
OC (9-16)	Fm7	Edim7	Fm7	Bb7	Eb6	Eb6	Eb6	Eb6
OR	Fm7	E9	Fm7	Bb7	Eb6	Eb6	Eb6	Eb6
NH	Edim7	Gm7	Fm7	Bb7	Eb6	Eb6	Eb6	Eb6

Rows beginning with OC correspond to the original chord progression. Rows beginning with OR correspond to the most likely chord structures given the original root note progression and melody with respect to the model presented in Section 4.5. Finally, rows beginning with NH correspond to a new harmonization generated by the same model and the root note progression model presented in Section 4.3 when observing original melody only.

gression, we see that most of the predicted chords are the same as the originals. When the chord differs, the musician will observe that the predicted chords are still relevant and are not in conflict with the original chords. It is more interesting to look at the sequence of chords generated by taking the sequence of root notes with the highest probability given by the root note progression model presented in Section 4.3 and then finding the most likely chord structures given this predicted root note progression and the original melody. While some chord changes are debatable, most of the chords comply with the melody and we think that the final result is musically interesting. These results show that valid harmonization models for melodies that could learn different musical styles could be implemented in commercial software in the short term. More generated results from the models presented in this chapter and audio examples are available at `http://www.idiap.ch/probmusic`.

4.6 Conclusion

To achieve appropriate harmonization, we introduced a chord decomposition that allows us to easily introduce domain knowledge in a probabilistic model by considering every structural component in chord notation. As in Chapter 2, we have shown empirically that chord progressions exhibit global

dependencies that can be better captured with a tree structure related to the meter than with a simple dynamical model that concentrates on local dependencies. However, the proposed local model seems to be sufficient when root note progressions are provided. This behavior suggests that most of the time-dependent information may already be contained in root note progressions.

Future work could be concerned with implementing the described approach in more general contexts. Repeating the tree structures over time could allow to generate harmonizations for melodies with arbitrary lengths. Our approach could also be trivially extended to classical chord notation.

Finally, the harmonization model introduced in this chapter can be sampled to generate realistic accompaniments to given melodies in the same style as a training corpus. This could have considerable interest in computer music software.

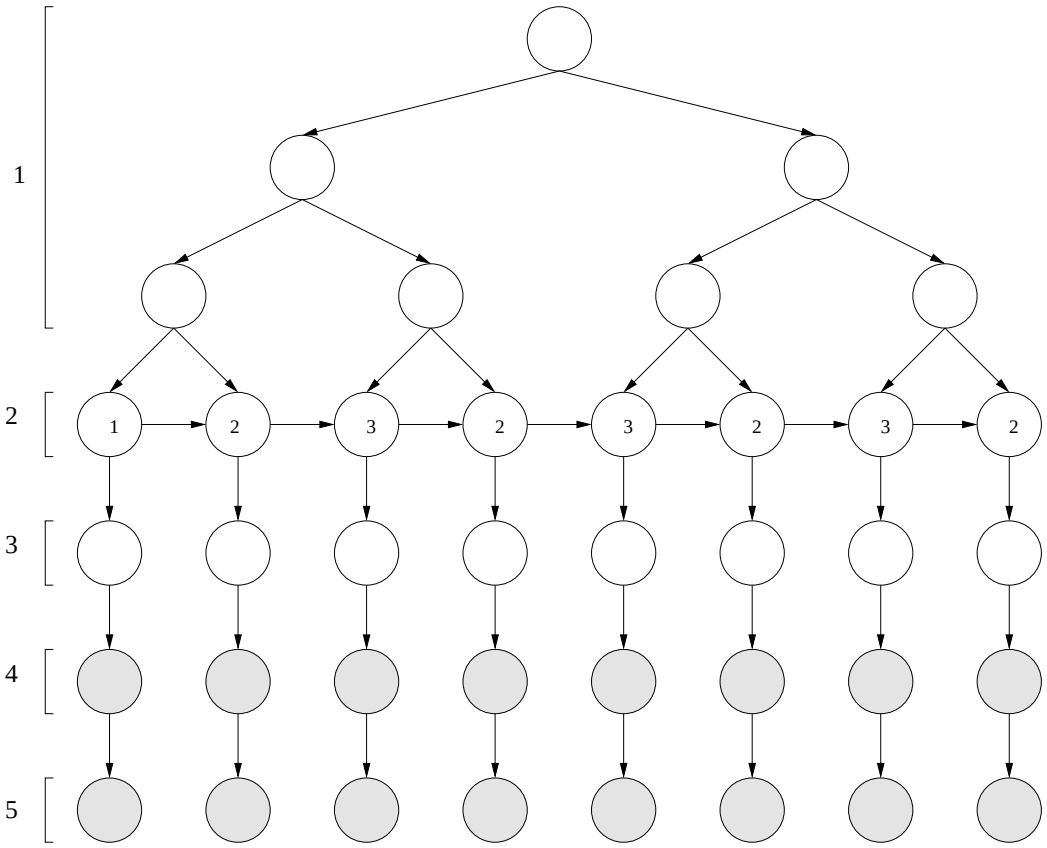

Figure 4.1. A graphical model to predict root note progressions given melodies. White nodes are hidden random variables while gray nodes are observed.

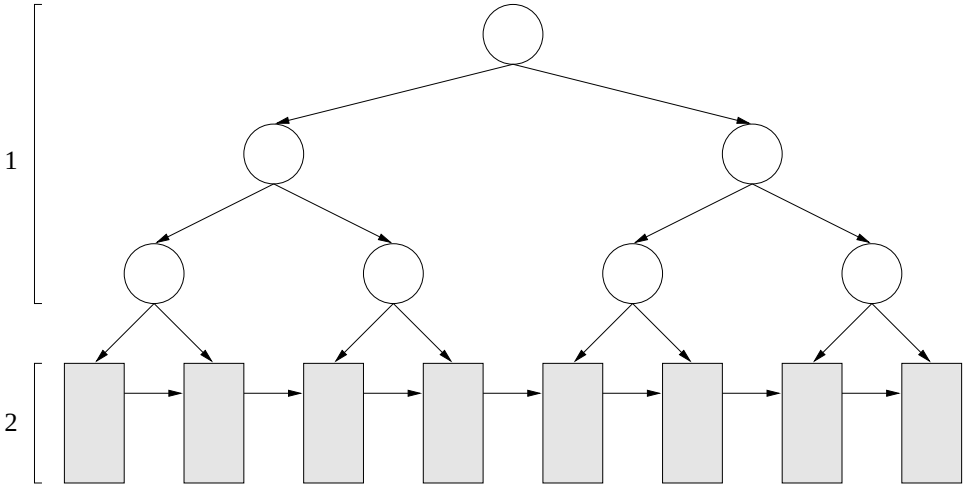

Figure 4.2. A graphical model to predict chord progressions given root notes progressions and melodies. The gray boxes correspond to subgraphs presented in Figure 4.3.

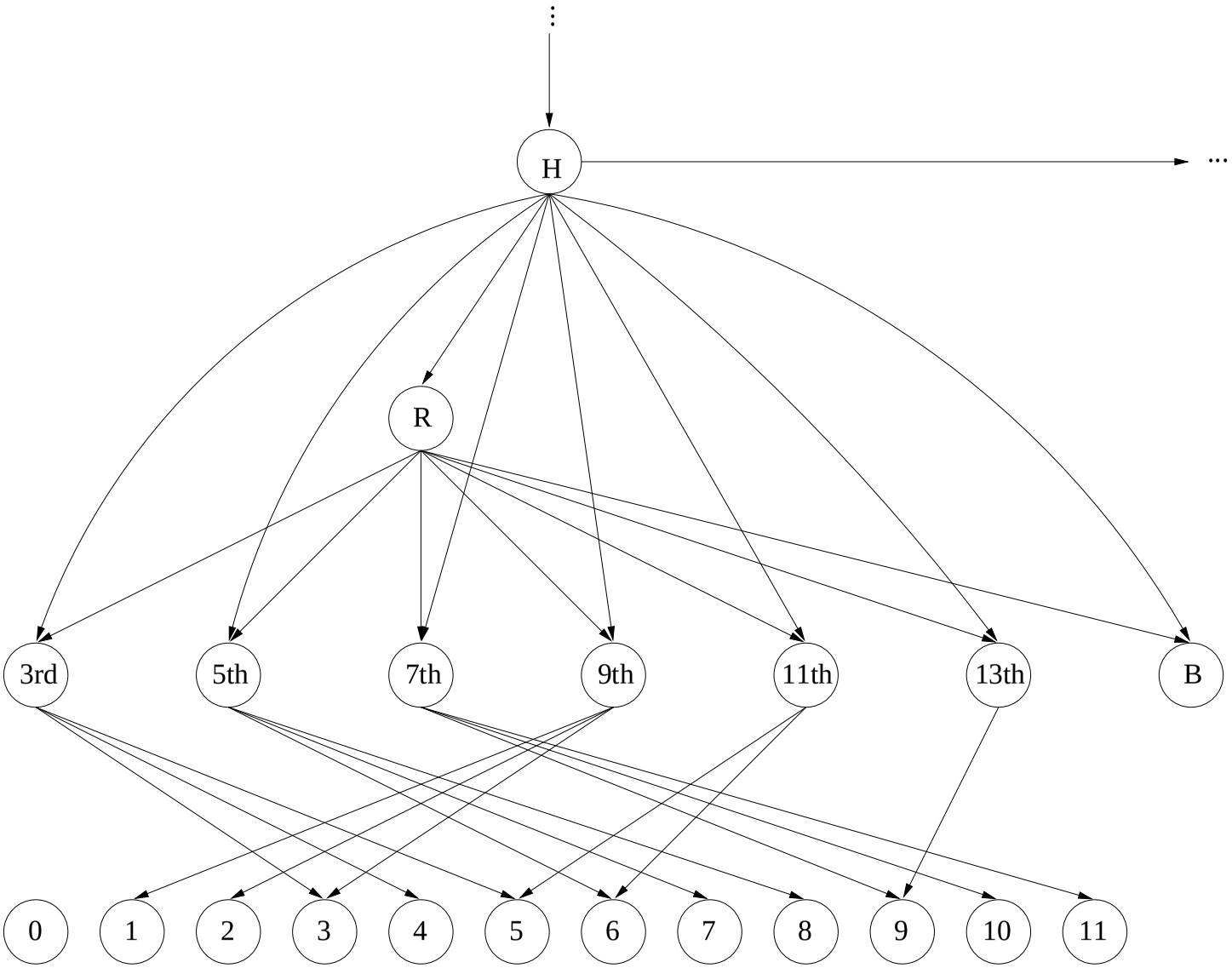

Figure 4.3. Subgraph of the graph presented in Figure 4.2. Each chord component is linked with the proper melodic components on the bottom.

5 Rhythms

The research reported in this chapter will be published in Paiement et al. [2008c].

As already stated in Section 1.2, music is characterized by strong hierarchical dependencies determined in large part by *meter*, the sense of strong and weak beats that arises from the interaction among hierarchical levels of sequences having nested periodic components. For example, a long melody is often composed by repeating with variation shorter sequences that fit into the metrical hierarchy (e.g. sequences of 4, 8 or 16 measures). In fact even random music can sound structured and melodic if it is built by repeating and varying random subsequences.

In this chapter, we focus on modeling rhythmic sequences, ignoring for the moment other aspects of music such as pitch, timbre and dynamics. Many algorithms have been proposed for audio beat tracking [Scheirer, 1998; Dixon, 2007]. Here, we consider rhythm modeling as a first step towards full melodic modeling. Our main contribution in this respect is to propose a generative model for distance patterns, specifically designed for capturing long-term dependencies in rhythms. In this work, distance patterns refer to distances between subsequences of equal length in particular positions. In Section 5.2, we describe the model, detail its implementation and present an algorithm using this model for rhythm prediction. The algorithm solves a constrained optimization problem, where the distance model is used to filter out rhythms that do not comply with the inferred structure. The proposed model is evaluated in terms of conditional prediction error on two distinct databases in Section 5.3, and a discussion follows.

We want to model rhythms in a dataset $\mathcal{X}$ consisting of rhythms of the same musical genre. We first quantize the database by segmenting each song

in m time steps and associate each note to the nearest time step, such that all melodies have the same length m. Note that this hypothesis is not fundamental in the proposed model and could easily be avoided if one would have to deal with more general datasets. It is then possible to represent rhythms by sequences containing potentially three different symbols: 1) Note onset, 2) Note continuation, and 3) Silence. When using quantization, there is a one to one mapping between this representation and the set of all possible rhythms. Using this representation, symbol 2 can never follow symbol 3. Let $A = \{1, 2, 3\}$; in the remaining of this chapter, we assume that $\mathbf{x}^l \in A^m$ for all $\mathbf{x}^l \in \mathcal{X}$.

5.1 HMMs for Rhythms

Let $\mathcal{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ be a dataset of rhythm sequences, where all the sequences contain m elements: $\mathbf{x}^l = (x_1^l, \ldots, x_m^l)$, $l = 1, \ldots, n$. Furthermore, let $\mathbf{h}^l = (h_1^l, \ldots, h_m^l)$ be the corresponding sequence of states for a discrete hidden variable synchronized with $\mathbf{x}^l$. The joint probability of the rhythm sequence $\mathbf{x}^l$ and hidden states $\mathbf{h}^l$ estimated by an HMM is given by

$$p_{\text{HMM}}(\mathbf{x}^l, \mathbf{h}^l) = p_\pi(h_1^l) p_o(x_1^l | h_1^l) \prod_{t=2}^{m} p_{\bar{o}}(h_t^l | h_{t-1}^l) p_o(x_t^l | h_t^l) \ , \tag{5.1}$$

where the $p_{\bar{o}}(.|.)$ terms are the transition probabilities, the $p_o(.|.)$ terms are the emission probabilities, and the $p_\pi(.)$ is the initial probability of the first state of the hidden variable. This model is presented in Figure 5.1, following standard graphical model formalism, where each node is associated to a random variable and arrows denote conditional dependencies. The probability distributions p_π, $p_{\bar{o}}$, and p_o are multinomials, whose parameters can be learned as usual by the EM algorithm

As stated in Section 1.3.2, HMMs are commonly used to model temporal data [Rabiner, 1989]. In principle, an HMM is able to capture complex regularities in patterns between subsequences of data, provided its number of hidden states is large enough. Thus, the HMM could be seen as a valid candidate for rhythm prediction. However, when dealing with rhythms in music, such a model would lead to a learning process requiring a prohibitive amount of data: in order to learn long range interactions, the training set should be representative of the joint distribution of subsequences. To overcome this problem, we propose in Section 5.2 to summarize the joint distribution of subsequences by the distribution of their pairwise distances. This summary is clearly not a sufficient statistic for the distribution of subsequences, but its distribution can be learned from a limited number of examples. The resulting model, which

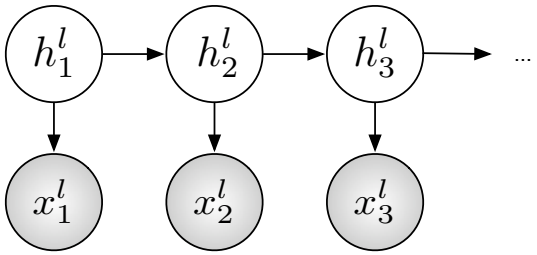

Figure 5.1. Hidden Markov Model. Each node is associated to a random variable and arrows denote conditional dependencies. When learning the parameters of the model, white nodes are hidden whereas grey nodes are observed.

generates distances, is then used to constrain the generation of subsequences. Moreover, empirical results obtained in Section 5.3 show that constraining the HMM with distributions over distance between subsequences significantly improves prediction accuracy.

5.2 Distance Model

As stated in Section 1.2, music is characterized by strong hierarchical dependencies determined in large part by *meter*, the sense of strong and weak beats that arises from the interaction among hierarchical levels of sequences having nested periodic components. Such a hierarchy is implied in western music notation, where different levels are indicated by kinds of notes (whole notes, half notes, quarter notes, etc.) and where bars establish measures of an equal number of beats. Meter and rhythm provide a framework for developing musical melody. For example, a long melody is often composed by repeating with variation shorter sequences that fit into the metrical hierarchy (e.g. sequences of 4, 8 or 16 measures). It is well known in music theory that *distance patterns* are more important than the actual choice of notes in order to create coherent music [Handel, 1993]. For instance, measure 1 may always be similar to measure 5 in a particular musical genre. In fact, even random music can sound structured and melodic if it is built by repeating random subsequences with slight variation.

Traditionally, musicologists refer to repetition patterns in music with sequences of letters (e.g. AABA). Let us consider the simple pattern "AB". This notation does not tell to what extent the second part differs from the first. Instead of just stating if the second part is similar or not to the first one, we want

to quantify the distances between the two parts in a corpus of music data. We can even go further and repeat this process hierarchically with various partition lengths. To do so, we introduce in this section a generative model for distance patterns and its application to rhythm sequences. Such a model is appropriate for most music data, where distances between subsequences of data exhibit strong regularities.

Suppose that we construct a *partition* of each rhythm sequence $\mathbf{x}^l$ by dividing it into ρ parts defined by $y_i^l = (x^l_{1+(i-1)m/\rho}, \ldots, x^l_{im/\rho})$ with $i \in \{1, \ldots, \rho\}$. We are interested in modeling the distances between these subsequences, given a suitable metric $d(y_i, y_j) : \mathbb{R}^{m/\rho} \times \mathbb{R}^{m/\rho} \to \mathbb{R}$. As we just pointed out, the distribution of $d(y_i, y_j)$ for each specific choice of i and j may be more important when modeling rhythms (and music in general) than the actual choice of subsequences y_i.

Let $D(\mathbf{x}^l) = (d^l_{i,j})_{1\leq i\leq\rho, 1\leq j\leq\rho}$ be the distance matrix associated with each sequence $\mathbf{x}^l$, where $d^l_{i,j} = d(y^l_i, y^l_j)$. Since $D(\mathbf{x}^l)$ is symmetric and contains only zeros on the diagonal, it is completely characterized by the upper triangular matrix of distances *without* the diagonal. Hence,

$$p(D(\mathbf{x}^l)) = \prod_{i=1}^{\rho-1} \prod_{j=i+1}^{\rho} p(d^l_{i,j}|S_{l,i,j}) \tag{5.2}$$

where

$$S_{l,i,j} = \{d^l_{r,s}|(1 < s < j \text{ and } 1 \leq r < s) \text{ or } (s = j \text{ and } 1 \leq r < i)\} \ . \tag{5.3}$$

In words, we order the elements column-wise and do a standard factorization, where each random variable depends on the previous elements in the ordering. Hence, we do not assume any conditional independence between the distances.

Since $d(y_i, y_j)$ is a metric, we have that $d(y_i, y_j) \leq d(y_i, y_k) + d(y_k, y_j)$ for all $i, j, k \in \{1, \ldots, \rho\}$. This inequality is usually referred to as the *triangle inequality*. Defining

$$\begin{aligned} \alpha^l_{i,j} &= \min_{k\in\{1,\ldots,(i-1)\}} (d^l_{k,j} + d^l_{i,k}) \text{ and} \\ \beta^l_{i,j} &= \max_{k\in\{1,\ldots,(i-1)\}} (|d^l_{k,j} - d^l_{i,k}|) \ , \end{aligned} \tag{5.4}$$

we know that given previously observed (or sampled) distances, constraints imposed by the triangle inequality on $d^l_{i,j}$ are simply

$$\beta^l_{i,j} \leq d^l_{i,j} \leq \alpha^l_{i,j} \ . \tag{5.5}$$

One may observe that the boundaries given in Eq. (5.4) contain a subset of the distances that are on the conditioning side of each factor in Eq. (5.2) for

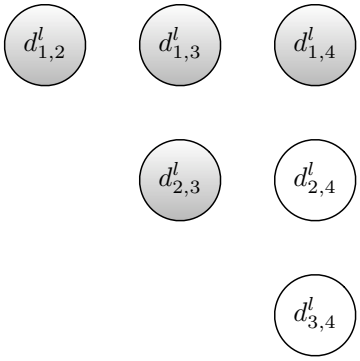

Figure 5.2. Each circle represents the random variable associated with the corresponding factor in Eq. (5.2), when $\rho = 4$. For instance, the conditional distribution for $d^l_{2,4}$ possibly depends on the variables associated to the grey circles.

each indices i and j. Thus, constraints imposed by the triangle inequality can be taken into account when modeling each factor of $p(D(\mathbf{x}^l))$: each $d^l_{i,j}$ must lie in the interval imposed by previously observed/sampled distances given in Eq. (5.5). Figure 5.2 shows an example where $\rho = 4$. Using Eq. (5.2), the distribution of $d^l_{2,4}$ would be conditioned on $d^l_{1,2}$, $d^l_{1,3}$, $d^l_{2,3}$, and $d^l_{1,4}$, and Eq. (5.5) reads $|d^l_{1,2} - d^l_{1,4}| \leq d^l_{2,4} \leq d^l_{1,2} + d^l_{1,4}$. Then, if subsequences y^l_1 and y^l_2 are close and y^l_1 and y^l_4 are also close, we know that y^l_2 and y^l_4 cannot be far. Conversely, if subsequences y^l_1 and y^l_2 are far and y^l_1 and y^l_4 are close, we know that y^l_2 and y^l_4 cannot be close.

5.2.1 Modeling Relative Distances Between Rhythms

When using the rhythm representation introduced in the beginning of this chapter, $d^l_{i,j}$ can simply be chosen to be the Hamming distance (i.e. counting the number of positions on which corresponding symbols are different). One could think of using more general edit distance such as the Levenshtein distance. However, this approach would not make sense psycho-acoustically: doing an insertion or a deletion in a rhythm produces a translation that alters dramatically the nature of the sequence. Putting it another way, rhythm perception heavily depends on the *position* on which rhythmic events occur. In the remainder of this chapter, $d^l_{i,j}$ is the Hamming distance between subsequences y_i and y_j.

We now have to encode our belief that rhythms of the same musical genre

have a common distance structure. For instance, drum beats in rock music can be very repetitive, except in the endings of every four measures, without regard to the actual beats being played. This should be accounted for in the distributions of the corresponding $d^l_{i,j}$.

With Hamming distances, the conditional distributions of $d^l_{i,j}$ in Eq. (5.2) should be modeled by discrete distributions, whose range of possible values *must* obey Eq. (5.5). Hence, we assume that the random variables $(d^l_{i,j} - \beta^l_{i,j})/(\alpha^l_{i,j} - \beta^l_{i,j})$ should be identically distributed for $l = 1, \ldots, n$. Empirical inspection of data supports this assumption. As an example, suppose that measures 1 and 4 always tend to be far away, that measures 1 and 3 are close, and that measures 3 and 4 are close; Triangle inequality states that 1 and 4 should be close in this case, but the desired model would still favor a solution with the greatest distance complying with the constraints imposed by triangle inequalities.

All these requirements are fulfilled if we model $d_{i,j} - \beta_{i,j}$ by a binomial distribution of parameters $(\alpha_{i,j} - \beta_{i,j}, p_{i,j})$, where $p_{i,j}$ is the probability that two symbols of subsequences y_i and y_j differ. With this choice, the conditional probability of getting $d_{i,j} = \beta_{i,j} + \delta$ would be

$$B(\delta, \alpha_{i,j}, \beta_{i,j}, p_{i,j}) = \binom{\alpha_{i,j} - \beta_{i,j}}{\delta} (p_{i,j})^{\delta} (1 - p_{i,j})^{(\alpha_{i,j} - \beta_{i,j} - \delta)} \quad , \tag{5.6}$$

with $0 \leq p_{i,j} \leq 1$. If $p_{i,j}$ is close to zero/one, the relative distance between subsequences y_i and y_j is small/large. However, the binomial distribution is not flexible enough since there is no indication that the distribution of $d_{i,j} - \beta_{i,j}$ is unimodal. We thus model each $d_{i,j} - \beta_{i,j}$ with a binomial *mixture* distribution in order to allow multiple modes. We thus use

$$p(d_{i,j} = \beta_{i,j} + \delta | S_{i,j}) = \sum_{k=1}^{c} w^{(k)}_{i,j} B(\delta, \alpha_{i,j}, \beta_{i,j}, p^{(k)}_{i,j}) \tag{5.7}$$

with $w^{(k)}_{i,j} \geq 0$, $\sum_{k=1}^{c} w^{(k)}_{i,j} = 1$ for every indices i and j, and $S_{i,j}$ defined similarly as in Eq. (5.3). Parameters

$$\theta_{i,j} = \{w^{(1)}_{i,j}, \ldots, w^{(c-1)}_{i,j}\} \cup \{p^{(1)}_{i,j}, \ldots, p^{(c)}_{i,j}\}$$

can be learned with the EM algorithm (c.f. Section 1.3.1) on rhythm data for a specific music style.

In words, we model the *difference* between the observed distance $d^l_{i,j}$ between two subsequences and the minimum possible value $\beta_{i,j}$ for such a difference by a binomial mixture.

The parameters $\theta_{i,j}$ can be initialized to arbitrary values before applying the EM algorithm. However, as the likelihood of mixture models is not a

convex function, one may get better models and speed up the learning process by choosing sensible values for the initial parameters. In the experiments reported in Section 5.3, the k-means algorithm for clustering [Duda et al., 2000a] was used. More precisely, k-means was used to partition the values $(d^l_{i,j} - \beta^l_{i,j})/(\alpha^l_{i,j} - \beta^l_{i,j})$ into c clusters corresponding to each component of the mixture in Eq. (5.7). Let $\{\mu^{(1)}_{i,j}, \ldots, \mu^{(c)}_{i,j}\}$ be the centroids and $\{n^{(1)}_{i,j}, \ldots, n^{(c)}_{i,j}\}$ the number of elements in each of these clusters. We initialize the parameters $\theta_{i,j}$ with

$$w^{(k)}_{i,j} = \frac{n^{(k)}_{i,j}}{n} \quad \text{and} \quad p^{(k)}_{i,j} = \mu^{(k)}_{i,j}.$$

We then follow a standard approach [Bilmes, 1997] to apply the EM algorithm to the binomial mixture in Eq. (5.7). Let $z^l_{i,j} \in \{1, \ldots, c\}$ be a hidden variable telling which component density generated $d^l_{i,j}$. For every iteration of the EM algorithm, we first compute

$$p(z^l_{i,j} = k | d^l_{i,j}, \alpha^l_{i,j}, \beta^l_{i,j}, \hat{\theta}_{i,j}) = \frac{\psi_{k,i,j,l}}{\sum_{t=1}^{c} \psi_{t,i,j,l}}$$

where $\hat{\theta}_{i,j}$ are the parameters estimated in the previous iteration, or the parameters guessed with k-means on the first iteration of EM, and

$$\psi_{k,i,j,l} = \hat{w}^{(k)}_{i,j} B(d^l_{i,j}, \alpha^l_{i,j}, \beta^l_{i,j}, p^{(k)}_{i,j}) \ .$$

Then, the parameters can be updated with

$$p^{(k)}_{i,j} = \frac{\sum_{l=1}^{n} (d^l_{i,j} - \beta^l_{i,j}) p(z^l_{i,j} = k | d^l_{i,j}, \alpha^l_{i,j}, \beta^l_{i,j}, \hat{\theta}_{i,j})}{\sum_{l=1}^{n} (\alpha^l_{i,j} - \beta^l_{i,j}) p(z^l_{i,j} = k | d^l_{i,j}, \alpha^l_{i,j}, \beta^l_{i,j}, \hat{\theta}_{i,j})}$$

and

$$w^{(k)}_{i,j} = \frac{1}{n} \sum_{l=1}^{n} p(z^l_{i,j} = k | d^l_{i,j}, \alpha^l_{i,j}, \beta^l_{i,j}, \hat{\theta}_{i,j}).$$

This process is repeated until convergence.

Note that using mixture models for discrete data is known to lead to *identifiability* problems. Identifiability refers here to the uniqueness of the representation (up to an irrelevant permutation of parameters) of any distribution that can be modeled by a mixture.

Estimation procedures may not be well-defined and asymptotic theory may not hold if a model is not identifiable. However, the model defined in Eq. (5.7) is identifiable if $\alpha_{i,j} - \beta_{i,j} > 2c - 1$ [Titterington et al., 1985, p.40]. While this is the case for most $d_{i,j}$, we observed that this condition is sometimes violated. Whatever happens, there is no impact on the estimation because we only care about what happens at the distribution level: there may be several parameters

leading to the same distribution, some components may vanish in the fitting process, but this is easily remedied, and EM behaves well.

As pointed out in Section 1.2, musical patterns form hierarchical structures closely related to meter [Handel, 1993]. Thus, the distribution of $p(D(\mathbf{x}^l))$ can be computed for many numbers of partitions within each rhythmic sequence. Let $\mathcal{P} = \{\rho_1, \dots \rho_h\}$ be a set of numbers of partitions to be considered by our model, where h is the number of such numbers of partitions. The choice of $\mathcal{P}$ depends on the domain of application. Following meter, $\mathcal{P}$ may have dyadic tree-like structure when modeling most music genres (e.g. $\mathcal{P} = \{2, 4, 8, 16\}$). Even when considering non-dyadic measures (e.g. a three-beat waltz), the *very large* majority of the hierarchical levels in metric structures follow dyadic patterns in most tonal music [Handel, 1993]. Let $D_{\rho_r}(\mathbf{x}^l)$ be the distance matrix associated with sequence $\mathbf{x}^l$ divided into ρ_r parts. Estimating the joint probability $\prod_{r=1}^{h} p(D_{\rho_r}(\mathbf{x}^l))$ with the EM algorithm as described in this section leads to a model of the distance structures in rhythms datasets. Suppose we consider 16 bar songs with four beats per bar. Using $\mathcal{P} = \{8, 16\}$ would mean that we consider pairs of distances between every group of two measures ($\rho = 8$), and every single measures ($\rho = 16$).

One may argue that our proposed model for long-term dependencies is rather unorthodox. However, simpler models like Poisson or Bernoulli process (we are working in discrete time) defined over the whole sequence would not be flexible enough to represent the particular long-term structures in music.

5.2.2 Conditional Prediction

For most music applications, it would be particularly helpful to know which rhythm sequence $\hat{x}_s, \dots, \hat{x}_m$ maximizes $p(\hat{x}_s, \dots, \hat{x}_m | x_1, \dots, x_{s-1})$. Knowing which musical events are the most likely given the past $s - 1$ observations would be useful both for prediction and generation. Note that in this thesis, we refer to prediction of musical events given past observations only for notational simplicity. All the generative models presented in this thesis could be used to predict any part of a music sequence given any other part with only minor modifications.

While the described modeling approach captures long range interactions in the music signal, it has two shortcomings. First, it does not model local dependencies: it does not predict how the distances in the smallest subsequences (i.e. with length smaller than $m/\max(\mathcal{P})$) are distributed on the events contained in these subsequences. Second, as the mapping from sequences to distances is many to one, there exists several admissible sequences $\mathbf{x}^l$ for a given set of

distances. These limitations are addressed by using another sequence learner designed to capture short-term dependencies between musical events. Here, we use an HMM for rhythms, as described in Section 5.1.

The two models are trained separately using their respective version of the EM algorithm. For predicting the continuation of new sequences, they are combined by choosing the sequence that is most likely according to the local HMM model, provided it is also plausible regarding the model of long-term dependencies. Let $p_{\text{HMM}}(\mathbf{x}^l)$ be the probability of observing sequence $\mathbf{x}^l$ estimated by the HMM after training. The final predicted sequence is the solution of the following optimization problem:

$$\begin{cases} \max\limits_{\tilde{x}_s,\ldots,\tilde{x}_m} & p_{\text{HMM}}(\tilde{x}_s,\ldots,\tilde{x}_m|x_1,\ldots,x_{s-1}) \\ \text{subject to} & \prod\limits_{r=1}^{h} p(D_{\rho_r}(\mathbf{x}^l)) \geq P_0 \ , \end{cases} \tag{5.8}$$

where P_0 is a threshold. In practice, one solves a Lagrangian formulation of problem (5.8), where we use log-probabilities for computational reasons:

$$\max_{\tilde{x}_s,\ldots,\tilde{x}_m} \left[\log p_{\text{HMM}}(\tilde{x}_s,\ldots,\tilde{x}_m|x_1,\ldots,x_{s-1}) + \lambda \sum_{r=1}^{h} \log p(D_{\rho_r}(\mathbf{x}^l))\right] \ , \tag{5.9}$$

where tuning λ has the same effect as choosing a threshold P_0 in Eq. (5.8) and can be done by cross-validation.

Multidimensional Scaling (MDS) is an algorithm that tries to embed points (here "local" subsequences) into a potentially lower dimensional space while trying to be faithful to the pairwise affinities given by a "global" distance matrix. Here, we propose to consider the prediction problem as finding sequences that maximize the likelihood of a "local" model of subsequences under the constraints imposed by a "global" generative model of distances between subsequences. In other words, solving problem (5.8) is similar to finding points such that their pairwise distances are as close as possible to a given set of distances (i.e. minimizing a stress function in MDS). Naively trying all possible subsequences to maximize (5.9) leads to $O(|A|^{(m-s+1)})$ computations, where $A = \{1, 2, 3\}$ is the set of rhythm symbols. Instead, we propose to search the space of sequences using a variant of the Greedy Max Cut (GMC) method [Rohde, 2002] that has proven to be optimal in terms of running time and performance for binary MDS optimization.

The subsequence $\hat{x}_s, \ldots, \hat{x}_m$ can be simply initialized with

$$(\hat{x}_s,\ldots,\hat{x}_m) = \max_{\tilde{x}_s,\ldots,\tilde{x}_m} p_{\text{HMM}}(\tilde{x}_s,\ldots,\tilde{x}_m|x_1,\ldots,x_{s-1}) \tag{5.10}$$

1. Initialize $\hat{x}_s, \ldots, \hat{x}_m$ using Eq. (5.10);
2. Set $j = s$ and set `end` = `true`;
3. Set $\hat{x}_j = \arg\max_{a \in A} [\log p_{\text{HMM}}(\mathbf{x}^* | x_1, \ldots, x_{s-1}) + \lambda \sum_{r=1}^{h} \log p(D_{\rho_r}(x_1, \ldots, x_{s-1}, \mathbf{x}^*))]$
   where $\mathbf{x}^* = (\hat{x}_s, \ldots, \hat{x}_{j-1}, a, \hat{x}_{j+1}, \ldots, \hat{x}_m)$
4. If $\hat{x}_j$ has been modified in the last step, set `end` = `false`.
5. If $j = m$ and `end` = `false`, go to 2;
6. If $j < m$, set $j = j + 1$ and go to 3;
7. Return $\hat{x}_s, \ldots, \hat{x}_m$.

Figure 5.3. Simple optimization algorithm to maximize $p(\hat{x}_s, \ldots, \hat{x}_m | x_1, \ldots, x_{s-1})$

using the local HMM model. The complete optimization algorithm is described in Figure 5.3. For each position, we try every admissible symbol of the alphabet and test if a change increases the probability of the sequence. We stop when no further change can increase the value of the utility function. Obviously, many other methods could have been used to search the space of possible sequences $\hat{x}_s, \ldots, \hat{x}_m$, such as simulated annealing [Kirkpatrick et al., 1983]. Our choice is motivated by simplicity and the fact that it yields excellent results, as reported in the following section.

5.3 Rhythm Prediction Experiments

Two databases from different musical genres were used to evaluate the proposed model. Firstly, jazz standards melodies [Sher, 1988] corresponding the the chord progressions described in Section 2.2.2 were interpreted and recorded by the first author in MIDI format. Appropriate rhythmic representations, as described in the beginning of this chapter, have been extracted from these files. Again, the complexity of the rhythm sequences found in this corpus is representative of the complexity of common jazz and pop music. We used the last 16 bars of each song to train the models, with four beats per bar. Two rhythmic observations were made for each beat, yielding observed sequences of length 128. We also used a subset of the Nottingham database (`http://www.cs.nott.ac.uk/~ef/music/database.htm`) consisting of 53 tra-

ditional British folk dance tunes called "hornpipes". In this case, we used the first 16 bars of each song to train the models, with four beats per bar. Three rhythmic observations were made for each beat, yielding observed sequences of length 192. The sequences from this second database contain no silence (or rest), leading to sequences with binary states.

Using similar notation as in Equation (3.1), let $x_t^{(n,i)}$ be the rhythmic observation in sequence n of the test set associated to the i-th fold of the cross-validation process at time t. Assume also that the i-th fold in the cross-validation process contains N_i test sequences, that there is a total of I folds in the cross-validation process and that all the sequences have length m. The prediction accuracy is given by

$$\frac{1}{I}\sum_{i=1}^{I}\frac{1}{N_i}\sum_{n=1}^{N_i}\frac{1}{m-s+1}\sum_{t=s}^{m}d_t^{(n,i)} \tag{5.11}$$

where

$$d_t^{(n,i)} = \begin{cases} 1 & \text{if } (\max_{x\in A} p_t^{(n,i)}(x|x_1^{(n,i)},\ldots,x_{s-1}^{(n,i)})) = x_t^{(n,i)} \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

with A being the set of possible rhythmic observations and $p_t^{(n,i)}(x|x_1^{n,i},\ldots,x_{s-1}^{(n,i)})$ being the conditional probability of observing rhythmic value x in sequence n of the test set associated to the i-th fold of the cross-validation process at time t, estimated by the evaluated model, given observations from time 1 through $s-1$. In words, the prediction accuracy is just the average number of times the algorithm makes a good choice when trying to predict rhythmic observations over songs *unseen* during training, given some previous observations. This performance criterion is qualitatively similar to the prediction error described in Equation (3.1), but considering a different context.

Note that, while the prediction accuracy is simple to estimate and to interpret, other performance criteria, such as ratings provided by a panel of experts, should be more appropriate to evaluate the relevance of music models. We plan to define such an evaluation protocol in future work.

Again, we used 5-fold double cross-validation to estimate the accuracies. Double cross-validation is a recursive application of cross-validation that enables to jointly optimize the hyper-parameters of the model and evaluate its generalization performance, as described in Section 2.2.2.

For the baseline HMM model, double cross-validation optimizes the number of possible states for the hidden variables. 2 to 20 possible states were tried in the reported experiments. In the case of the model with distance constraints, referred to as the global model, the hyper-parameters that were optimized are

the number of possible states for hidden variables in the local HMM model (i.e. 2 to 20), the Lagrange multiplier λ, the number of components c (common to all distances) for each binomial mixture, and the choice of $\mathcal{P}$, i.e. which partitions of the sequences to consider. Values of λ ranging between 0.1 and 4 and values of c ranging between 2 and 5 were tried during double cross-validation. Since music data commonly shows strong dyadic structure following meter, many subsets of $\mathcal{P} = \{2, 4, 8, 16\}$ were allowed during double cross-validation.

Note that the baseline HMM model is a poor benchmark on this task, since the predicted sequence, when prediction consists in choosing the most probable subsequence given previous observations, only depends on the state of the hidden variable at time $s-1$. This observation implies that the number of possible states for the hidden variables of the HMM upper-bounds the number of different sequences that the HMM can predict. However, this behavior of the HMM does not question the validity of the reported experiments. The main goal of this quantitative study is to measure to what extent distance patterns are present in music data and how well these dependencies can be captured by the proposed model. What we really want to measure is how much gain we observe in terms of out-of-sample prediction accuracy when using an arbitrary model if we impose additional constraints based on distance patterns. That being said, it would be interesting to measure the effect of appending distance constraints to more complex music prediction models [Dubnov et al., 2003; Pachet, 2003].

Results in Table 5.1 for the jazz standards database show that considering distance patterns significantly improves the HMM model. One can observe that the baseline HMM model performs much better when trying to predict the last 32 symbols. This is due to the fact that this database contains song endings. Such endings contain many silences and, in terms of accuracy, a useless model predicting silence at any position performs already well. On the other hand, the endings are generally different from the rest of the rhythm structures, thus harming the performance of the global model when just trying to predict the last 32 symbols. Results in Table 5.2 for the hornpipes database again show that the prediction accuracy of the global model is consistently better than the prediction accuracy of the HMM, but the difference is less marked. This is mainly due to the fact that this dataset only contains two symbols, associated to note onset and note continuation. Moreover, the frequency of these symbols is quite unbalanced, making the HMM model much more accurate when almost always predicting the most common symbol.

In Table 5.3, the set of partitions $\mathcal{P}$ is not optimized by double cross-

Table 5.1. Accuracy (the higher the better) for best models on the jazz standards database.

Observed	Predicted	HMM	Global
32	96	34.5%	54.6%
64	64	34.5%	55.6%
96	32	41.6%	47.2%

Table 5.2. Accuracy (the higher the better) for best models on the hornpipes database.

Observed	Predicted	HMM	Global
48	144	75.1%	83.0%
96	96	75.6%	82.1%
144	48	76.6%	80.1%

validation. Results are shown for different fixed sets of partitions. The best results are reached with "deeper" dyadic structure. This is a good indication that the basic hypothesis underlying the proposed model is well-suited to music data, namely that dyadic distance patterns exhibit strong regularities in music data. We did not compute accuracies for $\rho > 16$ because it makes no sense to estimate distribution of distances between too short subsequences.

Table 5.3. Accuracy over the last 64 positions for many sets of partitions $\mathcal{P}$ on the jazz database, given the first 64 observations. The higher the better.

$\mathcal{P}$	Global
$\{2\}$	49.3%
$\{2, 4\}$	49.3%
$\{2, 4, 8\}$	51.4%
$\{2, 4, 8, 16\}$	55.6%

5.4 Conclusion

The main contribution of this chapter is the design and evaluation of a generative model for distance patterns in temporal data. The model is specifically well-suited to rhythm data, which exhibits strong regularities in dyadic distance patterns. Reported conditional prediction accuracies show that the proposed model effectively captures such regularities. Rhythm prediction can be seen as the first step towards full melodic prediction and generation. In Chapter 6, we apply the proposed model to melody prediction. It could also be readily used to increase the performance of beat tracking algorithms, transcription algorithms, genre classifiers, or even automatic composition systems.

In future work, the distance constraints should be applied to other models than the HMM. Also, different initializations could be compared, as well as alternative optimization techniques. For instance, we could initialize the second half of a song with the first half. A greedy algorithm could then iteratively make modifications to the second half to introduce variability *where* is it usually found in the training corpus. In this case, a drum beat would have various "fill-ins" in the appropriate positions, while always keeping the same basic rhythmic structure anywhere else.

Finally, besides being fundamental in music, modeling distance patterns should also be useful in other application domains where data is represented as sequences, such as in natural language processing. Being able to characterize and constrain the relative distances between various parts of a sequence of bags-of-concepts could be an efficient means to improve performance of automatic systems such as machine translation [Och and Ney, 2004].

6 Melodies

With a reliable rhythm model available, we can turn our attention towards probabilistic modeling of melodies *given* rhythms and chord progressions. As was mentioned in Section 1.2, knowing the relations between chords and actual notes would certainly help to discover long-term musical structures in tonal music.

As we demonstrated in Chapter 4, it is fairly straightforward to generate interesting chord progressions given melodies in a particular musical genre [Allan and Williams, 2004; Paiement et al., 2006]. However, the dual problem that we address in this chapter is much more difficult. In Section 6.2.2, we describe melodic features derived from Narmour [1990] that put useful constraints on melodies based on musicological substantiation. We then introduce in Section 6.2.3 a probabilistic model of melodies *given* chords and rhythms that leads to significantly higher prediction rates than a simpler Markovian model. The combination of the rhythm model presented in Section 5.2 and the melodic model given chords of Section 6.2.3 leads to a predictive model of music that could be interesting in many applications.

The research reported in this chapter is currently under revision for publication in Paiement et al. [2008b].

6.1 Previous Work

We are not aware of existing generative models of melodies, *given* chords and rhythms, as we present in this chapter. However, we first describe previous works dealing with probabilistic modeling of polyphonic music in general. The presented models are not designed to extract melodies from rhythms and chords, but still involve various interactions between symbolic musical notes.

6.1.1 Bayesian Music Transcription

Cemgil et al. [2006] proposed a graphical model for audio to symbolic transcription.This model takes as input audio data, without any form of preprocessing. Instead of using Fourier transforms like in most current transcription systems, they model each harmonics of musical notes as sets of random variables. While being very costly, this approach has the advantage of being completely data-dependent. However, strong Markovian assumptions are necessary in order to model the temporal dependencies between notes, given the high complexity of the transcription graphical model.

Modeling a Single Note

Let $y_{1:T} = \{y_1 \ldots, t_T\}$ be a sequence of audio samples with constant frequency F_S.

First suppose that $y_{1:T}$ is a Gaussian process where typical realizations are damped sinusoids through time, with angular frequency ω:

$$\begin{aligned} s_t &\sim \mathcal{N}(\rho_t B(\omega) s_{t-1}, Q) \\ y_t &\sim \mathcal{N}(C s_t, R) \\ s_0 &\sim \mathcal{N}(0, S) \end{aligned}$$

where $\mathcal{N}(\mu, \Sigma)$ denotes a multivariate Gaussian distribution with mean μ and covariance Σ. Hence, Q, R, and S are appropriate covariance matrices. Here

$$B(\omega) = \begin{pmatrix} \cos(\omega) & -\sin(\omega) \\ \sin(\omega) & \cos(\omega) \end{pmatrix}$$

is a given rotation matrix that rotates two dimensional vector s_t by ω degrees counterclockwise. $C = [1,\ 0]$ is a projection matrix. The phase and amplitude of y_t are determined by the initial condition s_0 drawn from the prior with covariance S. The damping factor $0 \leq \rho_t \leq 1$ specifies the rate at which s_t contracts to 0.

Musical instruments have several modes of oscillation that are roughly located at multiple integer of the fundamental frequency ω. Cemgil et al. [2006] propose to model such signals by a bank of oscillators giving a block diagonal transition matrix $A(\omega, \rho_t)$ defined as

$$\begin{pmatrix} \rho_t^{(1)} B(\omega) & 0 & \cdots & 0 \\ 0 & \rho_t^{(2)} B(2\omega) & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \rho_t^{(H)} B(H\omega) \end{pmatrix}$$

where H is the given number of harmonics. This model can be intuitively seen as a probabilistic version of the Fourier decomposition of an observed waveform.

From audio to symbolic representation

Hidden variables for multiples notes onsets and offsets can be linked to the audio variables described so far. Let $r_{j,t}$ be indicator variables, where $j \in \{1, \ldots, M\}$ can be any possible note while $t \in \{1, \ldots, T\}$ runs over time. Each $r_{j,t}$ is binary with values `sound` or `mute`. Each note has a fundamental frequency ω_j. A polyphonic song can be seen as a collection of indicators $r_{1:M,1:T}$. Each row $r_{j,1:T}$ can be seen as controlling a sound generator as described in the last section. Then, Cemgil et al. [2006] describe a complex graphical model to define $p(r_{1:M,1:T}|y_{1:T})$, involving very strong Markovian assumptions about the prior transition probabilities $p(r_{1:M,1:T})$ between notes.

For transcription of an audio file $y_{1:T}$, the goal is to find to most likely polyphonic sequence of notes $r^*_{1:M,1:T}$ given by

$$\text{argmax}_{1:M,1:T} p(r_{1:M,1:T}|y_{1:T}) \ .$$

Given the high complexity of the model, this maximization is intractable and greedy approximation methods have to be used to estimate the most likely sequence of notes given an arbitrary audio sequence $y_{1:T}$. Learning the parameters in this model is even more difficult and requires approximation methods as well.

While this model is a very interesting modeling of transcription, we don't know the effect of using an overly simplistic model of $p(r_{1:M,1:T})$ on (poor) practical transcription performance. In this chapter, we propose a more realistic model for transitions between notes. We show empirically that the prediction accuracy of such a model is significantly better than what is obtained with a simpler Markovian model, as used in Cemgil et al. [2006].

6.1.2 Dictionary-based Predictors

Machine learning techniques have already been applied to music generation. In Dubnov et al. [2003], two distinct methods are proposed to generate melodies close to a given corpus. Both methods are *dictionary-based* predictors. A dictionary-based prediction method parses an existing musical text into a lexicon of phrases/patterns, called motifs, and provides an inference method for choosing the next musical object following a current past context.

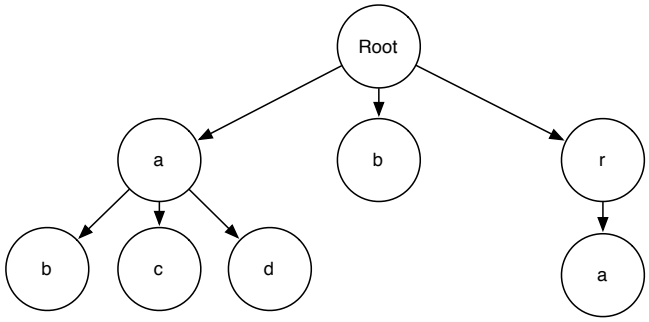

Figure 6.1. Dictionary tree created by IP when parsing the single sequence "abracadabra".

Incremental Parsing

An *incremental parsing* (IP) algorithm [Ziv and Lempel, 1978] builds a dictionary of distinct motifs by one continuous left to right traversal of a sequence, incrementally adding to a dictionary *every* new phrase that differs by a single last character from the longest match that already exists in the dictionary. For instance, given a text $\{ababaa...\}$, IP would parse $\{a, b, ab, aa, \ldots\}$. The dictionary may be efficiently represented by a tree, where each node correspond to an element of the dictionary.

Let $\mathcal{A}$ be an alphabet. For a node labeled by string c, the conditional probabilities of the descendents are taken as the relative portion of counts of characters $N_c(x),\ x \in \mathcal{A}$ appearing in all descendants of the current node c, setting

$$P_c(x) = \frac{N_c(x)}{\sum_{y \in \mathcal{A}} N_c(y)} \ .$$

As an example, the dictionary tree in Figure 6.1 is created when parsing a single sequence "abracadabra". In this context, the probability of generating "abrac" estimated by IP is

$$P(\text{"abrac"}) = P(a|\text{""})P(b|\mathbf{a})P(r|ab)P(a|ab\mathbf{r})P(c|abr\mathbf{a}) = 4/7{\cdot}1/3{\cdot}2/7{\cdot}1{\cdot}1/3 \ .$$

Prediction Suffix Trees (PST)

In contrast to the lossless coding scheme underlying IP, the PST algorithm [Ron et al., 1996] builds a restricted dictionary of *only* those motifs that both appear a significant number of times throughout the complete source sequence, *and* are meaningful for predicting the immediate future.

The empirical probability of a motif is the number of occurrences divided by the total number of occurrences. The conditional probability of a motif after a given context is the number of occurrences of that motif after the given context, divided by the number of occurrences of this context.

PST uses a breadth first search to find all the motifs that comply with the following requirements:

- The motif is no longer than some maximal length L;
- Its empirical probability is greater than P_{min};
- The conditional probability it induces over the next symbol differs by a multiplicative factor of at least r from that of the shorter contexts it subsumes, for at least one such next symbol.

Using this framework, the empirical probability of the motif "aa" within the sequence "aabaaab" is 3/6. The conditional probability of seeing "b" after "a" is 2/5. Also, the conditional probability of observing "b" after "aa" is 2/3. Finally, the multiplicative factor between the conditional probability to observe "b" after "aa" and that after "a" is $(2/3)/(2/5) = 5/3$.

In PST, the counts are incorporated using

$$\hat{P}_c(x) = (1 - |\mathcal{A}|g)P_c(x) + g$$

where $|\mathcal{A}|$ is the size of the alphabet and g is a smoothing factor, $0 < g < 1/|\mathcal{A}|$.

Melodic models based on IP and PST generate subjectively impressive musical results when sampled. However, we are not aware of musicological evidence to support such modeling of melodic sequences. In contrast, we propose in Section 6.2.2 a set of constraints on melodic shapes based on the musicological work of Narmour [1990].

Moreover, using generative models allows to generate musical events given appropriate contexts, which is less straightforward when using a dictionary-based approach. For instance, we propose a generative model of melodies *given* chords and rhythms in Section 6.2.3.

Finally, while models of music learned on data such as Espi et al. [2007]; Dubnov et al. [2003]; Pachet [2003] generate somewhat realistic musical results, we are only aware of a few quantitative comparisons between generative models of music [Lavrenko and Pickens, 2003], that is for instance in terms of out-of-sample prediction accuracy, as it is done in Sections 5.3 and 6.3.

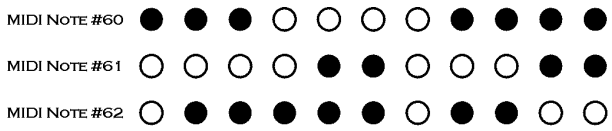

Figure 6.2. Representation of a polyphonic MIDI file, with millisecond time along the x-axis, and MIDI note number along the y-axis

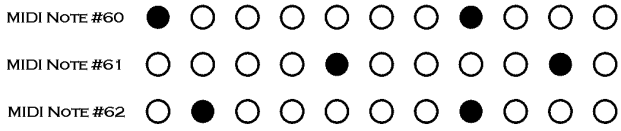

Figure 6.3. Simplification of the representation in Figure 6.2, where offset information is discarded, keeping only onset information. Black circles correspond to note onsets, with millisecond time along the x-axis, and MIDI note number along the y-axis.

6.1.3 Polyphonic Music Modeling with Random Fields

Lavrenko and Pickens [2003] propose a generative model of polyphonic music that employs Markov random fields (MRFs). Intuitively, an MRF is just a graphical model where each random variable can only depend on its close neighbors. However, very strong simplifications of the MIDI representation (with information loss) are necessary for the MRFs to be tractable when modeling polyphonic music.

A polyphonic MIDI file could be represented as a two dimensional MRF with millisecond time along the x-axis, and MIDI note number (from 1 to 128) along the y-axis, as in Figure 6.2. At any point along the y-axis, notes turn "on", remain "on" for a particular duration, and then turn back "off" again. From this accurate representation of MIDI data, Lavrenko and Pickens [2003] first discard the offset information. This leads to a representation as in Figure 6.3. Then, note durations and durations between notes are thrown away, as illustrated in Figure 6.4. The chronological relations between musical events is kept in this representation (including simultaneity of musical events). Finally, the 128-note y-axis is reduced to a 12-note pitch-class set, leading to the representation shown in Figure 6.5. As one can see, all these simplifications in the observed representations discard very important information present in

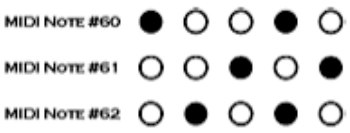

Figure 6.4. Simplification of the representation in Figure 6.3, where note durations and durations between notes are thrown away. Again, black circles correspond to note onsets. The chronological relations between musical event are kept in this representation.

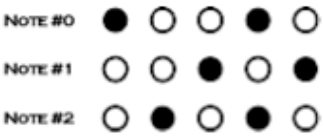

Figure 6.5. Simplification of the representation in Figure 6.4, where octave invariance is applied. MIDI note number information is converted to pitch-class information.

the original MIDI files. For instance, one could not directly sample generative models of this representation to generate polyphonic music. However, Lavrenko and Pickens [2003] argue that all these simplifications are necessary for their proposed MRFs to be tractable.

For each binary note $n_{i,t}$ with pitch-class i at time t, the goal is to estimate $P(n_{i,t}|H_{i,t})$, with

$$H_{i,t} = \{n_{j,s}|s < t\} \cup \{n_{j,s}|s = t, j < i\} \ .$$

Binary *feature functions* can have the form

$$f_S(n_{i,t}, H_{i,t}) = n_{i,t} \prod_{n_{j,s} \in S} n_{j,s}$$

where $S \subset H_{i,t}$ is made of close neighbors of $n_{i,t}$.

A possible parameterization of $P(n_{i,t}|H_{i,t})$ is defined by

$$\hat{P}(n_{i,t}|H_{i,t}) = \frac{1}{Z_{i,t}} \exp\left\{\sum_{f \in \mathcal{F}} \lambda_f f(n_{i,t}, H_{i,t})\right\}$$

where the λ_f are Lagrange multipliers, $Z_{i,t}$ is a normalization constant, and $\mathcal{F}$ is a set of feature functions. It is possible to learn the parameterization of

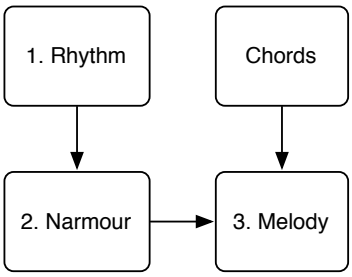

Figure 6.6. Schematic overview of the proposed melodic model. We first model rhythms. Then, we model Narmour features sequences given rhythms. Finally, we model actual melodies given Narmour features and chords.

$\hat{P}(n_{i,t}|H_{i,t})$ that maximizes the likelihood of training data. Pietra et al. [1997] describe how to learn the values of λ_f and the structure of $\mathcal{F}$ simultaneously for this purpose.

Lavrenko and Pickens [2003] show that using their MRF model for polyphonic music outperforms a simple Markov chain in terms of prediction accuracy. Though the MRFs model is not restricted to chord progressions, the dependencies it considers (by putting constraints on the choices of S) are much shorter than in the chord models presented in Chapter 2. Also, octave information and temporal information are discarded, making the model unsuitable for modeling realistic chord voicings or melodies. For instance, low notes tend to have more salience in chords than high notes [Levine, 1990]. While being very general, this model would benefit from having access to more specific musical knowledge, as it is the case for most of the models introduced in this thesis.

6.2 Melodic Model

We described a reliable prediction model for rhythms in Chapter 5. We can now turn our attention to an even more difficult problem, which is to model melody notes, *given* rhythm and chord progressions. In order to do so, we proceed iteratively in three steps depicted schematically in Figure 6.6. We first model rhythms with the model presented in Section 5.2. Then, we model features that represent the plausibility of sequences of notes. These "Narmour" features, introduced in Section 6.2.2, are computed for each sequence of three consecutive notes. Their prediction is an interesting intermediate problem since the cardinality of such features is much lower than the number of sequences

of three notes. Moreover, such features are descriptive of the perceptual expectancy of a particular group of three notes. As stated in Section 1.2, chords can be seen as latent variables (local in time) that condition the probabilities of choosing particular notes in a melody. However, chords do not describe longer term melodic structure. This is why we propose to use Narmour features as sequences of constraints on the choices of melody notes. In Section 6.2.3, we describe a probabilistic model for melody notes given Narmour features *and* chord progressions.

Results reported in Section 6.3 show that using sequences of Narmour features as constraints leads to much better prediction accuracy than the direct baseline approach using the IOHMM model described in the following section. We do not compare quantitatively our presented model with the polyphonic music models presented in Section 6.1, introduced in previous work. The main reason is that our proposed model is a model of melodies *given* chords and rhythms. On the other hand, the various models of polyphonic music presented in Section 6.1 are *unconditional* models of polyphonic music. However, the combination of our proposed conditional melodic model with the chord progression model of Chapter 2 and the rhythm model of Chapter 5 could form an unconditional polyphonic music model by itself. As such, it could be compared with previous probabilistic models of symbolic polyphonic music. We defer this comparison to future work.

6.2.1 IOHMMs

A simple probabilistic model for melodies given chords can be designed by adding input variables to the HMM, as it is done in Section 3.2.2. Let $\mathcal{U} = \{\mathbf{u}^1, \ldots, \mathbf{u}^n\}$ be a dataset of varying length melodies, where melody $\mathbf{u}^l$ has length g_l, $\mathbf{u}^l = (u^l_1, \ldots, u^l_{g_l})$. Each melodic line is composed of notes u^l_i in the MIDI standard, $u^l_i \in \{0, \ldots, 127\}$. The melodies in dataset $\mathcal{U}$ are synchronized with rhythms in the dataset $\mathcal{X}$ defined as in Section 5.2. The length g_l of melodic line $\mathbf{u}^l$ corresponds thus to the number of note onsets (symbol 1) in rhythm sequence $\mathbf{x}^l$. In addition, let $\nu^l = (\nu^l_1, \ldots, \nu^l_{g_l})$ be the chord progression corresponding to the l-th melody. In the experiments reported in this chapter, each ν^l_t is represented by the `Naive` representation described in Section 3.2.3. Hence, each ν^l_t takes a discrete value within the number of different chords in the dataset. The joint probability of each sequence $\mathbf{u}^l$, its associated chord

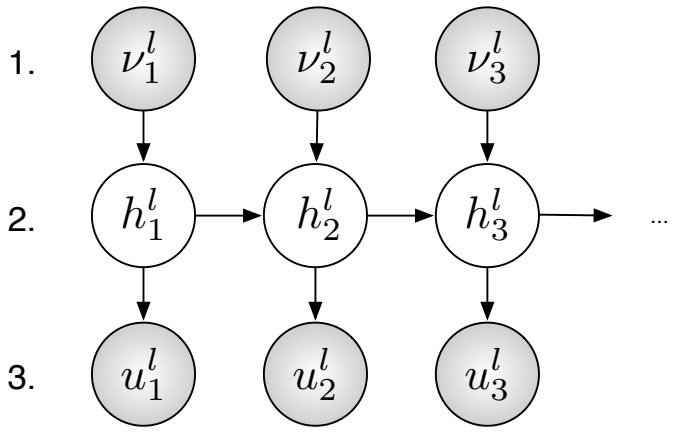

Figure 6.7. Variant of an IOHMM model for MIDI notes given chords. The variables in level 1 are always observed and correspond to chords. Variables in level 2 are hidden, while variables in level 3 correspond to melodic notes. All variables in grey are observed during training.

progression ν^l, and hidden states $\mathbf{h}^l$ can be modeled by

$$p_{\text{IOHMM}}(\mathbf{u}^l, \nu^l, \mathbf{h}^l) = p_i(\nu_1^l) p_\pi(h_1^l|\nu_1^l) p_o(u_1^l|h_1^l) \prod_{t=2}^{g_l} p_i(\nu_t^l) p_{\bar{o}}(h_t^l|h_{t-1}^l, \nu_t^l) p_o(u_t^l|h_t^l) \quad . \tag{6.1}$$

This model, shown in Figure 6.7, is a specific Input/Output Hidden Markov Model (IOHMM), as introduced by Bengio and Frasconi [1996]. Usual IOHMMs have additional links connecting directly the input variables (level 1) to the outputs (level 3). We removed these links to decrease to number of parameters in the model, and thus being less prone to overfit the training data.

The probability distributions p_π, p_i, $p_{\bar{o}}$, and p_o are multinomials, as in Equation (5.1), and the model is learned by the standard EM algorithm. Performance of the IOHMM in terms of melodic prediction accuracy given chords is presented in Section 6.3.

6.2.2 Narmour Features

In this section, we introduce melodic features that will prove to be useful for melodic prediction. The Implication-Realization (I-R) model has been developed by Narmour [1990, 1992] as a theory of musical expectation. This fairly complex musicological model was then simplified and implemented by Schellenberg [1997], who proposed a formal analysis of each sequence of three consec-

utive notes, according to five perceptual items: registral direction, intervallic difference, registral return, proximity, and closure, as described later in this section. The model returns five scores measuring expectancy according to these five criteria, and, according to Narmour's theory, high perceptual expectancy incurs high cumulative scores. This model was empirically shown to be relevant in information retrieval applications [Grachten et al., 2005].

In this chapter, our goal is quite different. Instead of quantifying melodic expectancy, we design a probabilistic model of melodic sequences given chords. We propose to collectively use the Narmour principles as discrete features to characterize each sequence of three consecutive notes. In the remainder of this thesis, we refer to these features as *Narmour features*. There is far fewer possible Narmour features (108 in our implementation) than possible groups of three notes (128^3 if we consider all MIDI notes). Given that observation, we expect that modeling sequences of Narmour features should be easier than modeling actual sequences of notes. We describe in Section 6.2.3 how we propose to generate actual melodies given sequences of Narmour features.

Our particular implementation of the Narmour features is mostly derived from Schellenberg [1997]. We simply define the *interval* v_t between two notes u_t and u_{t-1} to be the difference $v_t = u_{t-1} - u_t$ between their MIDI note numbers. Interval has to be taken here in its musicological sense, which is not related to the usual mathematical definition: an interval is an integer that counts the number of semi-tones between two notes. Each Narmour principle can be computed for any sequence of three consecutive notes, corresponding to two intervals. In Narmour's theory, the first interval is referred to as the *Implication* while the second interval corresponds to the *Realization* of a melodic pattern of three notes. We define the sign function as

$$\text{sgn}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} .$$

The registral direction principle states that continuation in pitch direction is expected after small intervals and that large intervals imply a change of direction. We define

$$\text{rm}_t = \begin{cases} 0 & \text{if } |v_{t-1}| > 6 \text{ and } \text{sgn}(v_{t-1}) = \text{sgn}(v_t) \\ 1 & \text{if } |v_{t-1}| \leq 6 \\ 2 & \text{if } |v_{t-1}| > 6 \text{ and } \text{sgn}(v_{t-1}) \neq \text{sgn}(v_t) \end{cases}$$

to be the Narmour feature scoring the registral direction principle computed on arbitrary MIDI notes u_{t-2}, u_{t-1}, and u_t.

The intervallic difference principle says that small intervals imply similar-sized realized intervals and that large implicative intervals imply relatively smaller realized intervals. Formally,

$$\mathrm{id}_t = \begin{cases} 1 & \text{if } |v_{t-1}| < 6 \text{ and } \mathrm{sgn}(v_{t-1}) \neq \mathrm{sgn}(v_t) \text{ and } ||v_{t-1}| - |v_t|| < 3 \\ 1 & \text{if } |v_{t-1}| < 6 \text{ and } \mathrm{sgn}(v_{t-1}) = \mathrm{sgn}(v_t) \text{ and } ||v_{t-1}| - |v_t|| < 4 \\ 1 & \text{if } |v_{t-1}| > 6 \text{ and } |v_{t-1}| \geq |v_t| \\ 0 & \text{otherwise} \end{cases}$$

is the Narmour feature scoring the intervallic difference principle.

The registral return principle states that the second tone of a realized interval is expected to be very similar to the original pitch (within 2 semi-tones). Thus, we define the following scoring function

$$\mathrm{rr}_t = \begin{cases} 1 & \text{if } |v_t + v_{t-1}| \leq 2 \\ 0 & \text{otherwise.} \end{cases}$$

Then, the closure principle states that either melody changes direction, or that large intervals are followed by a relatively smaller interval. This feature is scored by

$$\mathrm{cl}_t = \begin{cases} 2 & \text{if } \mathrm{sgn}(v_{t-1}) \neq \mathrm{sgn}(v_t) \text{ and } |v_{t-1}| - |v_t| > 2 \\ 1 & \text{if } \mathrm{sgn}(v_{t-1}) \neq \mathrm{sgn}(v_t) \text{ and } |v_{t-1}| - |v_t| < 3 \\ 1 & \text{if } \mathrm{sgn}(v_{t-1}) = \mathrm{sgn}(v_t) \text{ and } |v_{t-1}| - |v_t| > 3 \\ 0 & \text{otherwise.} \end{cases}$$

Finally, the proximity principle favors small realized intervals. We define

$$\mathrm{pr}_t = \begin{cases} 0 & \text{if } |v_t| \geq 6 \\ 1 & \text{if } 3 \leq |v_t| \leq 5 \\ 2 & \text{if } 0 \leq |v_t| \leq 2 \end{cases} .$$

We define this feature with less possible states than in [Schellenberg, 1997] in order to limit the dimensionality of the Narmour representation. Besides, the actual numerical values for each of the Narmour features do not correspond to those of Schellenberg [1997], where the goal was to quantify numerically the subjective melodic expectation. In the context of this chapter, these values only correspond to discrete ordered values summarizing triplets of notes.

From these definitions, the Narmour features for the note triplet (u_{t-2}, u_{t-1}, u_t) are defined as

$$\gamma_t = (\mathrm{rm}_t, \mathrm{id}_t, \mathrm{rr}_t, \mathrm{cl}_t, \mathrm{pr}_t) \ .$$

Such features have 108 possible different discrete states.

As an example, the sequence of MIDI notes $(u_1, u_2, u_3, u_4) = (71, 74, 72, 84)$ would lead to the Narmour features $\gamma_3 = (1, 1, 1, 1, 2)$ and $\gamma_4 = (1, 0, 0, 1, 0)$.

6.2.3 Melodic Model

In this section, we describe a probabilistic model for melodies given rhythms and chord progressions. While the IOHMM in Section 6.2.1 was directly modeling the choice of notes given chords (and implicitly rhythms), the model described here proceeds in two steps. We first model sequences of Narmour features given rhythm. Then, we model the actual choice of melodic notes, given sequences of Narmour features generated in the last step and chord progressions. These two steps correspond to steps 2. and 3. in Figure 6.6.

IOHMM for Narmour Features

An IOHMM like the one presented in Section 6.2.1 can be used to model sequences of Narmour features given rhythms. We first compress the rhythm dataset in a form that is synchronized with Narmour features: we define $\mathbf{a}^l = (a^l_2, \ldots, a^l_{g_l-1})$ to be the l-th sequence of note lengths in the rhythm dataset $\mathcal{X}$, ignoring the first and last note lengths a^l_1 and $a^l_{g_l}$. When considering the rhythm representation defined in Section 5.2.1, each a^l_i is equal to one plus the number of symbols 2 following the i-th symbol 1 in the corresponding rhythm sequence $\mathbf{x}^l$. For instance, the rhythm sequence $\mathbf{x}^l = (1, 1, 2, 2, 3, 1, 2, 1)$ produces the note length sequence $\mathbf{a}^l = (3, 2)$. We denote by $\gamma^l = (\gamma^l_3, \ldots, \gamma^l_{g_l})$ the sequence of Narmour features associated to the l-th melody. This sequence starts with index 3 because each Narmour feature spans three notes.

The joint probability of each sequence of Narmour features γ^l, its associated sequence of note lengths $\mathbf{a}^l$, and hidden states $\mathbf{h}^l$ can be modeled by

$$\begin{aligned} p_{\text{NIOHMM}}(\mathbf{a}^l, \gamma^l, \mathbf{h}^l) \quad = \quad & p_i(a^l_2)p_\pi(h^l_1|a^l_2)p_o(\gamma^l_3|h^l_1) \\ & \prod_{t=4}^{g_l} p_i(a^l_{t-1})p_{\bar{o}}(h^l_{t-2}|h^l_{t-3}, a^l_{t-1})p_o(\gamma^l_t|h^l_{t-2}) \quad . \quad (6.2) \end{aligned}$$

This model is shown in Figure 6.8. As in Equation (6.1), the probability distributions p_π, p_i, $p_{\bar{o}}$, and p_o are multinomials, and the model is learned by the standard EM algorithm.

As can be seen in Equation (6.2), we arbitrarily chose to condition the Narmour features on the *previous* note length. This is due to the empirical observation that greater intervals tend to occur after long notes while smaller intervals tend to occur after short notes. Other models of Narmour features given *current* length, a longer past context, or even no note length at all could be considered. We leave this exploration for future work.

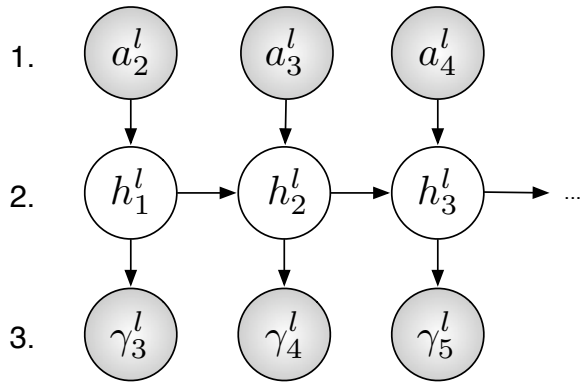

Figure 6.8. Variant of an IOHMM model for Narmour features given note lengths. The variables in level 1 are always observed and correspond to *previous* note lengths. Variables in level 2 are hidden, while variables in level 3 correspond to Narmour features. All variables in grey are observed during training.

Notes Model

We are now about to reach the end of our quest for a complete generative model of melodies given chord progressions. The only piece of the puzzle that remains to be defined is a model for MIDI notes given Narmour features and chord progressions.

We first decompose the `Naive` chord representation defined in Section 3.2.3 into two parts: $\nu_i^l = (\eta_i^l, \tau_i^l)$, where η_i^l is the structure of the chord and τ_i^l is the root pitch class. Chord structures are just the chord definitions aside of the name of the root (e.g. "m7b5" is the chord structure in the chord "Bm7b5"). Each different chord structure is mapped to a specific state of the variables η_i^l. The sequences $\eta^l = (\eta_1^l, \ldots, \eta_{g_l}^l)$ and $\tau^l = (\tau_1^l, \ldots, \tau_{g_l}^l)$ are respectively the *chord structure* and the *root* progressions of the l-th song in the dataset.

Let $\tilde{u}_t^l$ be an arbitrary MIDI note played at time t. We define

$$\phi(\tilde{u}_t^l, \tau_t^l) = ((\tilde{u}_t^l \mod 12) - \tau_t^l) \mod 12$$

to be the representation of the pitch class associated to the MIDI note $\tilde{u}_t^l$, *relative* to the root of the current chord. For instance, let $\tilde{u}_t^l = 65$ (note F) be played over the D minor chord. In that case, we have $\tau_t^l = 2$, meaning that the pitch class of the root of the chord is D. Hence, $\phi(65, 2) = 3$ for that particular example, meaning that the current melody note pitch class is 3 semi-tones higher than the root of the current chord.

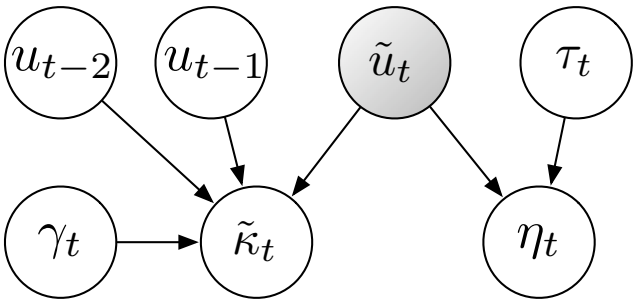

Figure 6.9. Graphical model representation of the factorization of the joint probability defined in Eq. (6.3).

It is easy to estimate $p(\eta_t^l|\tilde{u}_t^l, \tau_t^l)$ with a multinomial distribution conditioned on the values of $\phi(\tilde{u}_t^l, \tau_t^l)$. This distribution can be estimated by maximum likelihood over a training set. Hence, we learn a simple distribution of the chord structures η for each possible pitch classes of the melodies relative to the roots of the corresponding chords. For instance, this distribution could learn the fact that we often observe a minor seventh chord when playing a minor third over the tonic in the melody. This is somewhat similar to the melodic representation used in the harmonization model detailed in Figure 4.3, on page 66.

Let $\tilde{\gamma}_t^l(u_{t-2}^l, u_{t-1}^l, \tilde{u}_t^l)$ be the extracted Narmour feature when notes u_{t-2}^l and u_{t-1}^l are followed by the arbitrary note $\tilde{u}_t^l$. Also, let $\tilde{\kappa}_t^l$ be an arbitrary random variable such that

$$p(\tilde{\kappa}_t^l = 1|\tilde{u}_t^l, u_{t-2}^l, u_{t-1}^l, \gamma_t^l) = \begin{cases} 1 & \text{if } \gamma_t^l = \tilde{\gamma}_t^l(u_{t-2}^l, u_{t-1}^l, \tilde{u}_t^l) \\ 0 & \text{otherwise.} \end{cases}$$

In words, $\tilde{\kappa}_t^l$ is equal to 1 if and only if the Narmour feature produced when playing arbitrary note $\tilde{u}_t^l$ is equal to the *given* Narmour feature γ_t^l, given the two previous notes.

We define a factorization of the joint probability of the variables $\tilde{u}_t^l, u_{t-1}^l$, $u_{t-2}^l, \eta_t^l, \tau_t^l, \gamma_t^l$, and $\tilde{\kappa}_t^l$ with

$$\begin{aligned} &p(\tilde{u}_t^l, u_{t-1}^l, u_{t-2}^l, \eta_t^l, \tau_t^l, \gamma_t^l, \tilde{\kappa}_t^l) = \\ &\quad p(u_{t-1}^l)p(u_{t-2}^l)p(\gamma_t^l)p(\tilde{\kappa}_t^l|\tilde{u}_t^l, u_{t-2}^l, u_{t-1}^l, \gamma_t^l)p(\tilde{u}_t^l)p(\tau_t^l)p(\eta_t^l|\tilde{u}_t^l, \tau_t^l) \end{aligned} \tag{6.3}$$

at each time t. This factorization is is shown by the graphical model in Figure 6.9.

We want to estimate the probability of playing any arbitrary MIDI note $\tilde{u}_t^l$ at time t in the l-th song of the dataset given the two previous observed notes u_{t-2}^l and u_{t-1}^l, the current Narmour feature γ_t^l, and the current chord $\nu_t^l = (\eta_t^l, \tau_t^l)$. Given the factorization in Equation (6.3), we have that

$$p_{MEL}(\tilde{u}_t^l | u_{t-1}^l, u_{t-2}^l, \eta_t^l, \tau_t^l, \gamma_t^l, \tilde{\kappa}_t^l = 1) = \frac{p(\tilde{\kappa}_t^l=1|\tilde{u}_t^l, u_{t-2}^l, u_{t-1}^l, \gamma_t^l) p(\tilde{u}_t^l) p(\eta_t^l | \tilde{u}_t^l, \tau_t^l)}{\sum_{\tilde{u}_t^l} p(\tilde{\kappa}_t^l=1|\tilde{u}_t^l, u_{t-2}^l, u_{t-1}^l, \gamma_t^l) p(\tilde{u}_t^l) p(\eta_t^l | \tilde{u}_t^l, \tau_t^l)} \tag{6.4}$$

where $p(\tilde{u}_t^l)$ is the prior probability of observing $\tilde{u}_t^l$. The distribution $p(\tilde{u}_t^l)$ is a multinomial that can be simply estimated by maximum likelihood on the training set.

Hence, a simple strategy to find the most likely MIDI note $\tilde{u}_t^l$ given u_{t-1}^l, $u_{t-2}^l, \eta_t^l, \tau_t^l$, and γ^l is to solve

$$\arg \max_{\{\tilde{u}_t^l | \tilde{\kappa}_t^l = 1, u_{t-1}^l, u_{t-2}^l, \gamma^l\}} p(\tilde{u}_t^l) p(\eta_t^l | \tilde{u}_t^l, \tau_t^l) \ ,$$

since the denominator in the right-hand side of Equation (6.4) is the same for all values of $\tilde{u}_t^l$. In other words, we search for the most likely melodic note (with respect to the current chord) among all the possible notes given the current Narmour constraint and the current chord. Despite the fact that this model only predict one note at a time, it is able to take into account longer term melodic shapes through the constraints imposed by the sequences of Narmour features.

Melodic prediction *without* observing Narmour features can be done with this model in two steps. We first generate the most likely sequence of Narmour features given rhythms with the IOHMM model described in Section 6.2.3. Then, we can use the melodic prediction model described in the current section to predict MIDI notes given chord progressions. Such a model is shown in Section 6.3 to have much better prediction accuracy than using a simpler IOHMM model alone.

We show in Chapter 2 that unsupervised probabilistic models can be sampled to generate genuine chord progressions. The melodic model described here is able to generate realistic melodies given these chord progressions and beginning of melodies. This system can be used as a tool to ease music composition. Audio files generated by sampling the different models presented in this chapter are available at `http://www.idiap.ch/probmusic`. Even for the non musician, it should be obvious that the sequences generated by sampling the melodic model introduced in this section are much more realistic than sequences generated by sampling the IOHMM model described in Section 6.2.1.

Both models generate notes that are coherent with the current chord. However, the sequences generated by the IOHMM model do not have any coherent temporal structure. On the other hand, melodies generated by the melodic model presented here tend to follow the same melodic shapes than the songs in the training sets. These melodic shapes are constrained by the conditioning sequences of Narmour features used as inputs.

6.3 Melodic Prediction Experiments

To compare the melodic model described in the previous section with the IOHMM model of Section 6.2.1, we propose a slightly different evaluation criterion than the prediction accuracy defined in Section 5.3. This alternate criterion was chosen in this case for its computational simplicity.

The goal of the proposed models is to predict or generate melodies *given* chord progressions and rhythm patterns. Using similar notation as in Equation (3.1), let $\mathbf{u}^{(n,i)} = (u_1^{(n,i)}, \ldots, u_{g_{(n,i)}}^{(n,i)})$ be the test sequence of MIDI notes in sequence n of the test set associated to the i-th fold of the cross-validation process. Let $\nu^{(n,i)}$ be the sequence of chords associated to $\mathbf{u}^{(n,i)}$. Assume also that the i-th fold in the cross-validation process contains N_i test sequences, that there is a total of I folds in the cross-validation process and each sequence have length $g_{(n,i)}$. We define the *local accuracy* of an evaluated model to be

$$\frac{1}{I}\sum_{i=1}^{I}\frac{1}{N_i}\sum_{n=1}^{N_i}\frac{1}{g_{(n,i)}-s+1}\sum_{t=\zeta_s^{(n,i)}}^{g_{(n,i)}} d_t^{(n,i)} \tag{6.5}$$

where

$$d_t^{(n,i)} = \begin{cases} 1 & \text{if } (\max_{\tilde{u}_t^{(n,i)}\in\{0,\ldots,127\}} p_t^{(n,i)}(\tilde{u}_t^{(n,i)}|u_1^{(n,i)},\ldots,u_{t-1}^{(n,i)},\nu^{(n,i)})) = u_t^{(n,i)} \\ 0 & \text{otherwise} \end{cases} \tag{6.6}$$

with $p_t^{(n,i)}(\tilde{u}_t^{(n,i)}|u_1^{(n,i)},\ldots,u_{t-1}^{(n,i)},\nu^{(n,i)}))$ being the conditional probability of observing note $\tilde{u}_t^{(n,i)}$ in sequence n of the test set associated to the i-th fold of the cross-validation process at time t, estimated by the evaluated model. The variable $\zeta_s^{(n,i)}$ is the smallest note index in the same sequence, such that its corresponding rhythm index is equal or greater than s. In words, the prediction accuracy is just the average number of times the algorithm makes a good choice when trying to predict rhythmic observations over songs *unseen* during training, given *all* past observations and the whole chord progression. The difference between this performance criterion and the prediction accuracy described in Equation (5.11) is that the models now have access to all the previous notes, and the whole chord progressions, to make prediction.

Table 6.1. Local accuracy (the higher the better) for prediction models on the jazz standards database, for various prediction starting points s.

s	IOHMM	Narmour
32	2.0%	8.9%
64	1.7%	8.1%
96	2.2%	8.3%

Table 6.2. Local accuracy (the higher the better) for prediction models on the hornpipes database, for various prediction starting points s.

s	IOHMM	Narmour
48	2.5%	4.6%
96	2.6%	4.8%
144	2.6%	4.9%

2 to 20 possible hidden states were tried in the reported experiments for the baseline IOHMM model of Section 6.2.1 and the "Narmour" IOHMM of Section 6.2.3. Both models try to predict out-of-sample melody notes, given chord progressions and complete test rhythm sequences. The same chord representations are used as input for both models. 5-fold cross-validation was used to compute prediction accuracies. We report results for the choices of parameters that provided the highest accuracies for each model. The IOHMM model of notes given chords is a stronger contender than would be a simpler HMM trained on melodies, because the prediction given by the IOHMM takes advantage of the current input.

Results in Table 6.1 for the jazz standards database show that generating Narmour features as an intermediate step greatly improves prediction accuracy. Since there are 128 different MIDI notes, a completely random predictor would have a local accuracy of 0.8%. Both models take into account chord progressions when trying to predict the next MIDI note. However, the Narmour model favors melodic shapes similar to the ones found in the training set. The Narmour model still provides consistently better prediction accuracy on the hornpipes database, as can be seen in Table 6.2. However, prediction accuracies are lower on the hornpipes database than on the jazz database for

the Narmour model. Note onsets (symbol 1) occur on most rhythm positions in this database. This means that rhythm sequences in this database have relatively low entropy. Hence, rhythm sequences are less informative when used as conditioning inputs to generate sequences of Narmour features. Another observation is that the chord structures in this database are almost always the same (i.e. simple triads). The melodic model of Section 6.2.3 is directly modeling the distribution $p(\eta_t^l|\tilde{u}_t^l, \tau_t^l)$ of chord structures given relative MIDI notes. This distribution was probably more helpful for melodic prediction in the jazz database than in the hornpipes database. Despite these two drawbacks, the melodic model of Section 6.2.3 has a prediction accuracy twice as good as what was obtained with the simpler IOHMM model in the hornpipes database.

Again, while the prediction accuracy is simple to compute and to apprehend, other performance criteria, such as ratings provided by a panel of experts, should be more appropriate to evaluate the relevance of music models. The fact that the Narmour model accurately predict "only" about 8% of the notes on out-of-sample sequences does not mean that it is not performing well when generating the other "wrong" notes. Many realistic melodies can be generated on the same chord progression in a given musical genre. Moreover, some mistakes are more harmful than others. For most applications, a model that would have very low prediction accuracy, but that would generate realistic melodies, would be preferable to a model with 50% prediction accuracy, but that would generate unrealistic notes the other half of the time.

6.4 Conclusion

Our main contribution in this chapter is the design and evaluation of a realistic generative model for melodies. While a few models have already been proposed to generate music [Espi et al., 2007; Dubnov et al., 2003; Pachet, 2003], we are only aware of a few quantitative comparisons between limited generative models of music [Lavrenko and Pickens, 2003].

We defined in Section 6.2.3 a probabilistic model of melodies that provides significantly higher prediction rates than a simpler, yet powerful, Markovian model. It would be interesting to use the distributions on distances introduced in Chapter 5 to constrain the IOHMM of Narmour features. Also, future work could try to measure the prediction accuracy of Narmour features themselves. The combination of the rhythm model and the melodic model given chords leads to a computational model of music that could be interesting in many applications. Furthermore, sampling these models given appropriate musical contexts generates realistic melodies and rhythms. Many melodic models could

be combined to generate polyphonic music. One would have to design appropriate dependencies between all individual models.

7 Conclusion

In this thesis, we have shown empirically that it is possible to design *probabilistic models* that are able to *predict* and to *generate* music given arbitrary contexts in a genre similar to a training corpus, using a minimal amount of music data.

More specifically, our main contributions are three-fold:

- We have shown empirically that long term dependencies *are* present in music data and we provided quantitative measures of such dependencies;
- Our most important contribution is to have shown empirically that using domain knowledge allows to capture long term dependencies in music signal better than with standard statistical models for temporal data. We included domain knowledge in probabilistic models in three distinct ways:
  - We modeled global dependencies with probabilistic trees tied to the metrical structure of the datasets;
  - We introduced a generative model for pairwise distances between rhythm subsequences;
  - We modeled melodic features derived from musicological substantiation.

  Hence, we introduced many probabilistic models aimed to capture various aspects of symbolic polyphonic music. Such models can be used for music prediction. Moreover, these models can be sampled to generate realistic music sequences;
- Finally, we designed various representations for music that could be used as observations by the proposed probabilistic models.

7.1 Motivation

We first introduced basic musical concepts in Section 1.2. One of the key observations we made is that metrical grids provide a temporal framework around which a piece of music is organized. Also, chord progressions can be seen as latent variables that condition the probabilities of choosing particular notes. Chord changes occur on fixed time intervals strongly tied to the metrical grid. This makes them much simpler to model than beginning and ending of actual musical notes that can happen almost everywhere. Hence, chord progressions modeling appears to be a excellent way to overcome long term dependencies in polyphonic music.

We described the wide spectrum of possible computer music applications in Section 1.2.5. Most computer music research has been devoted so far to the direct modeling of audio data. These algorithms rely on local properties of audio signal, such as texture, or short term frequency analysis. Hence, most of the music models today *do not* consider the musical structure at all. Little research has been done to model symbolic music data compared to the important efforts deployed to model audio data. We argue that reliable symbolic music models such a the ones presented in this thesis could dramatically improve the performance of audio algorithms applied in more general contexts. This would be a very interesting avenue for future research. Reliable music models would provide "musical knowledge" to algorithms that currently only rely on basic sound properties to make decisions. In the same manner, natural language models are commonly used in speech transcription algorithms. Moreover, symbolic music data is much more compressed than audio data. We can concentrate on essential psychoacoustic features of the signal when designing algorithms to capture long term dependencies in symbolic music data.

We introduced elements of machine learning in Section 1.3. This thesis is mostly concerned with the design of *generative* models defined in the graphical model framework. Defining a graphical model representation for a set of random variables amounts to defining a set of *independence assumptions* between these variables, by factorization of their joint distribution. Also, the graphical modeling framework provides efficient tools for marginalization, such as the junction tree algorithm. Parameter learning can be usually done in simple graphical models with the EM algorithm. We described the HMM in Section 1.3.2 as a simple, but yet powerful, generative model for temporal data.

One of the advantages of generative models is that they can be sampled to generate new data from the learned distributions. Also, generative models can

be used to estimate the conditional probability of some random variables given new observations. For instance, a generative model can guess what would be a good accompaniment for a given melody, as in Chapter 4. Conversely, one could sample a generative model to generate realistic melodies given harmonic context, as in Chapter 6. Hence, generative models provide an ideal framework to model music.

A model with too few parameters would be unable to learn appropriate distributions properly. On the other hand, a model with too much parameters would learn the training set by heart and would fail to *generalize* to unseen data. Three approaches can be taken to overcome this problem: Build or collect more data, design better representations for data, and design better algorithms given *a priori* knowledge of the structure of data. These three approaches were considered in this thesis.

The few existing MIDI databases available today are severely limited in size. Moreover, they comprise only specific musical genres. A dataset of jazz standards (described in Section 2.2.2) was recorded by the author. This dataset is representative of the complexity of common jazz and pop music. It is used in the experiments reported in this thesis along with other public datasets.

A more important contribution of this thesis is the design of representations that exhibits important statistical properties of music data. In Chapter 2, we introduced a distributed representation for chords, such that Euclidean distances correspond to psychoacoustical similarities. In Chapter 3, we compared various chord representations quantitatively in terms of prediction accuracy. In Chapter 4, compressed representations of melodies and chords for harmonization were introduced. In the following chapters, simple representations for rhythms and melodies were also described as inputs to polyphonic music models. We finally described discrete representations of groups of three melodic notes based on musicological theory in Chapter 6. We showed that such representations could be modeled more easily than actual sequences of notes.

Having access to sufficiently large datasets and to reliable representations of these observations is the basis of any machine learning system. However, most of this thesis was concerned with the most important part of statistical analysis, which is the design and evaluation of algorithms themselves.

7.2 Chord models

In Section 2.2, we have shown empirically that chord progressions exhibit global dependencies that can be better captured with a tree structure related to the meter than with a simple dynamical HMM that concentrates on local

dependencies [Paiement et al., 2005b].

The low difference in terms of conditional out-of-sample likelihood between the tree model and the HMM, and the relatively low optimal capacity for generalization are a good indication that increasing the number of sequences in the datasets would probably be necessary in further developments of probabilistic models for chord progressions. Results reported in Section 2.3 provide further evidence that chord progressions exhibit global dependencies that follow hierarchical structure related to the meter [Paiement et al., 2005a]. We described in this section an alternative approach, where domain knowledge is included in the conditional probability distributions, instead of being included in the representation of the observations.

Applications where a chord progression model could be included range from music transcription, music information retrieval, musical genre recognition to music analysis applications, just to name a few.

Chord progressions are regular and simple structures that condition dramatically the actual choice of notes in polyphonic tonal music. Hence, we argue that chord models are crucial in the design of efficient algorithms that deal with such music data. Moreover, generating interesting chord progressions can be considered to be one of the most important aspects in generating realistic polyphonic music.

Chapter 3 [Paiement et al., 2008a] was aimed at better understanding the statistical relations between chord representations and the actual choice of notes in polyphonic music. To this end, we compared four chord representations using melodic prediction as a benchmark task. It should be pointed out that a similar approach could be taken in future research to evaluate representations for other musical components, such as rhythm or melodies.

The simplest chord representation (`Naive`) performs well in terms of unconditional prediction error rates. However, this representation overfits when past melodic observations are used to condition the predictions. Smoothed chord representations seems more appropriate when doing conditional prediction. Only observing the roots seems to be a good compromise when all the songs are transposed to the same key. We show in Section 4.3 that root progressions contain valuable non-local information. The representation that is used in some important music information retrieval papers (`Root+mM7`) is not optimal in any of the experiments. That being said, chord representation should always be chosen considering the application to be developed in mind.

We also observed in Chapter 3 that the evaluation provided by the average out-of-sample likelihood can be completely different from what is obtained when

computing the average prediction error rate. While the likelihood is a measure of the fit of a whole distribution to a dataset, the classification error seems to be a better descriptor of the fit of the modes of a distribution.

Chapter 4 [Paiement et al., 2006] was devoted to harmonization. This is a problem that has considerable practical interest. For instance, a good model for harmonization could be readily included in music sequencers. Amateur musicians could compose melodies and the system would be able to suggest realistic accompaniments in a style similar to a training corpus.

We introduced a chord decomposition specifically designed for harmonization. Again, we observed that chord progressions exhibit global dependencies that can be better captured with a tree structure related to the meter than with a simple dynamical model that concentrates on local dependencies. However, we made a very interesting observation related to root progressions. A local model seems to be sufficient when root note progressions are provided. Hence, most of the time-dependent information is contained in root note progressions, at least in the corpus that we used for the experiments.

7.3 Rhythms and Melodies

In Chapter 5 [Paiement et al., 2008c], we focused on the design and evaluation of a generative model for distance patterns in temporal data. We applied this model to rhythm data, which exhibits strong regularities in dyadic distance patterns. We have shown empirically that the proposed model effectively captures such regularities. Our proposed model could be readily used to increase the performance of beat tracking algorithms, transcription algorithms, genre classifiers, or even automatic composition systems.

Modeling distance patterns should also be useful in other application domains where data is represented as sequences, such as in natural language processing. For instance, being able to characterize and constrain the relative distances between various parts of a sequence of bags-of-concepts could be an efficient mean to improve machine translation.

Finally, in Chapter 6 [Paiement et al., 2008b], our main contribution was the design and evaluation of a realistic generative model for melodies. In Section 6.2.2, we described melodic features that put useful constraints on melodies based on musicological substantiation. In future research, other features could be used to put constraints on melodies.

It is easy to generate interesting accompaniments in various music styles given any chord progression. Hence, all the models presented in this thesis could be readily combined to generate genuine polyphonic music from basic melodic

motives. First, the harmonization model of Chapter 4 could generate a sequence of chords given an initial melodic motif. Then, one of the models presented in Chapter 2 could be sampled to extend the chord progression generated by the harmonization model. An extended rhythm could be generated as well from the initial motif by sampling the model introduced in Chapter 5. Finally, a whole melody could be generated by the melodic model of Chapter 6, given the already sampled chord progression and rhythms.

The primary contribution of this thesis is scientific in essence: we proposed a set of probabilistic models that have been shown empirically to better learn long-term dependencies in music data than with state-of-the-art machine learning approaches. However, one may argue that our most exciting contribution may be closer to the domain of arts than to the realm of science. We have described a set of models than can be used to generate realistic music in a musical genre similar to almost any training corpus of tonal music.

Bibliography

M. Allan and C. K. I. Williams. Harmonising chorales by probabilistic inference. In *Advances in Neural Information Processing Systems*, volume 17, 2004.

Jean-Julien Aucouturier and François Pachet. Representing musical genre: A state of the art. *Journal of New Music Research*, 32(1):83–93, 2003. NOTE: Classify representations of musical genre in three categories: manual, prescriptive and emergent approaches.

J. P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the Sixth International Conference on Music Information Retrieval*, pages p.304–311, London, 2005.

Y. Bengio and P. Frasconi. Input/output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, 1996.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2): 157–166, 1994.

Adam Berenzweig, Daniel P.W. Ellis, and Steve Lawrence. Anchor space for classification and similarity measurement of music. In *IEEE International Conference on Multimedia and Expo*, 2003. NOTE: Mapping of music into a space where dimensions correspond to "semantic" features.

P. Billingsley. *Probability and Measure*. John Wiley and Sons, New York, 1995.

J. Bilmes. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models, 1997. URL citeseer.ist.psu.edu/bilmes98gentle.html.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

L. Breiman. *Probability*. Addison-Wesley, Reading, MA, 1968.

Christopher J.C. Burges, John C. Platt, and Soumaya Jana. Distorsion discriminant analysis for audio fingerprinting. *IEEE Transactions on Speech and Audio Processing*, 11(3):165–174, 2003. NOTE: This method constructs a linear, convolutional neural network out of layers, each of which performs an oriented PCA.

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.

Ali Taylan Cemgil, Peter Desain, and Bert Kappen. Rhythm quantization for transcription. *Computer Music Journal*, 24(2):60–76, 2000. NOTE: Quantization of short groups of onsets using bayesian statistics.

Ali Taylan Cemgil, Bert Kappen, and David Barber. Generative model based polyphonic music transcription. In *Proc. of IEEE WASPAA*, New Paltz, NY, October 2003. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. NOTE: Dynamical bayesian network for simultaneous tempo and polyphonic pitch tracking.

Grosvenor Cooper and Leonard B. Meyer. *The Rhythmic Structure of Music*. The Univ. of Chicago Press, 1960.

Carl Dahlhaus. *Studies on the Origin of Harmonic Tonality*. Princeton University Press, 1990. XV-389 p.

M. Davy and S. J. Godsill. Bayesian harmonic models for musical signal analysis. *Bayesian Statistics*, 7, 2003. NOTE: Use of Bayesian structures in order to infer quantities about musical signals at the highest level, such as pitch, dynamics, timbre, instrument identity, etc.

Alain de Cheveigné and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *J. Acoust. Soc. Am.*, 111:1917–1930, 2002. NOTE: A fundamental frequency estimator based on autocorrelations.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

S. Dixon. Evaluation of the audio beat tracking system beatroot. *Journal of New Music Research*, 36(1):39–50, 2007.

S. Dubnov, G. Assayag, O. Lartillot, and G. Bejerano. Using machine-learning methods for musical style modeling. *IEEE Computer*, 10(38), October 2003.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, Second Edition*. Wiley Interscience, 2000a.

Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000b. ISBN 0471056693.

Douglas Eck and Juergen Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In H. Bourlard, editor, *Neural Networks for Signal Processing XII, Proc. 2002 IEEE Workshop*, pages 747–756, New York, 2002. IEEE.

D. Espi, P.J. Ponce de Leon, C. Perez-Sancho, D. Rizo, J.M. Inesta, F. Moreno-Seco, and A. Pertusa. A cooperative approach to style-oriented music composition. In *Proc. of the Int. Workshop on Artificial Intelligence and Music, MUSIC-AI*, pages 25–36, Hyderabad, India, 2007.

W. Feller. *An Introduction to Probability Theory and its Applications*. John Wiley and Sons, New York, 2nd edition edition, 1971.

Jonathan Foote. A similarity measure for automatic audio classification. In *Proceedings of the AAAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora*, March 1997. NOTE: Use of the MMI-supervised vector quantizer as a measure of audio similarity.

Masataka Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171, 2001. NOTE: Uses onset times, chord changes, and drum patterns.

Masataka Goto and Yoichi Muraoka. An audio-based real-time beat tracking system and its applications. In *Proceedings of the 1998 International Computer Music Conference*, pages 17–20, October 1998. NOTE: Real-time beat tracking system based on onset times, chord changes and drum patterns.

M. Grachten, J. LI. Arcos, and R. Lopez de Mantaras. Melody retrieval using the implication/realization model. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2005.

Stephen Handel. *Listening: An introduction to the perception of auditory events*. MIT Press, Cambridge, Mass., 1993.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer series in statistics. Springer-Verlag, 2001.

D. Heckerman, D. Geiger, and M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. Technical report, Microsoft Research, 1994.

B.-H. Juang and S. Katagiri. Discriminative learning for minimum error classification. In *IEEE Trans. on Signal Processing*, volume 10 (12), pages 3043–3053, 1992.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983. URL citeseer.ist.psu.edu/kirkpatrick83optimization.html.

Anssi Klapuri. Automatic transcription of music. Master's thesis, Tampere University of Technology, 2001.

Anssi Klapuri. Pitch estimation using multiple independent time-frequency windows. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, Oct. 17-20 1999. NOTE: Pitch model that calculates independent pitch estimates in separate time-frequency windows and then combines them.

Anssi Klapuri, Tuomas Virtanen, and Jan-Markus Holm. Robust multipitch estimation for the analysis and manipulation of polyphonic musical signals. In *Proc. COST-G6 Conference on Digital Audio Effects, DAFx-00*, Verona, Italy, 2000. NOTE: Pitch estimation on one time frame from chords comprising 1 to 6 distincts notes.

Carol M. Krumhansl. The psychological representation of musical pitch in a tonal context. *Cognitive Psychology*, 11(3):346–374, July 1979.

T. Kuusi. *Set-Class and Chord: Examining Connection Between Theoretical Ressemblance and Perceived Closeness*. Number 12 in Studia Musica. Sibelius Academy, 2001.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

V. Lavrenko and J. Pickens. Polyphonic music modeling with random fields. In *Proceedings of ACM Multimedia*, pages 120–129, Berkeley, CA, November 2-8 2003.

Mark Levine. *The Jazz Piano Book*. Sher Music Co./Advance Music, 1990.

Thomas Lidy, Andreas Rauber, Antonio Pertusa, and José Manuel Iñesta. Improving genre classification by combination of audio and symbolic descriptors using a transcription system. In *Proceedings of the 8th International Conference on Music Information Retrieval*, pages 23–27, Vienna, Austria, September 2007.

Keith D. Martin. Automatic transcription of simple polyphonic music. In *Third Joint Meeting of the Acoustical Societies of America and Japan*, 1996. NOTE: A transcription system based on the log-lag correlogram.

B.C.J. Moore. *An Introduction to the Psychology of Hearing*. Academic Press, 1982.

Eugene Narmour. *The Analysis and Cognition of Basic Melodic Structures: The Implication-Realization Model*. Univeristy of Chicago Press, Chicago, 1990.

Eugene Narmour. *The Analysis and Cognition of Melodic Complexity: The Implication-Realization Model*. University of Chicago Press, 1992.

F. J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

F. Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, September 2003.

F. Pachet and A. Zils. Evolving automatically high-level music descriptors from acoustic signals. *Springer Verlag LNCS*, 2771, 2003. NOTE: Genetic algorithms used to extract descriptors from audio signals.

François Pachet and Daniel Cazaly. A taxonomy of musical genres. In *Proceedings of Content-Based Multimedia Information Access (RIAO) Conference*, Paris, France, 2000. NOTE: Analysis of existing taxonomies of musical genre and description of a new one.

François Pachet, Gert Westermann, and Damien Laigre. Musical data mining for electronic music distribution. In *Proceedings of the 1st WedelMusic Conference*, 2001. NOTE: Classical data-mining techniques such as co-occurence and correlation analysis for classification of music titles.

J.-F. Paiement, D. Eck, and S. Bengio. A probabilistic model for chord progressions. In *Proceedings of the 6th International Conference on Music Information Retrieval*, 2005a.

J.-F. Paiement, D. Eck, S. Bengio, and D. Barber. A graphical model for chord progressions embedded in a psychoacoustic space. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005b.

J.-F. Paiement, D. Eck, and S. Bengio. Probabilistic melodic harmonization. In *Proceedings of the 19th Canadian Conference on Artificial Intelligence*, pages 218–229. Springer, 2006.

J.-F. Paiement, Samy Bengio, and Douglas Eck. Probabilistic models for melodic prediction. IDIAP-RR 08-50, Idiap Research Institute, 2008a. Currently under revision in *Artificial Intelligence*.

J.-F. Paiement, Y. Grandvalet, and S. Bengio. Predictive models for music. IDIAP-RR 08-51, Idiap Research Institute, 2008b. Currently under revision in *Connection Science*.

J.-F. Paiement, Y. Grandvalet, S. Bengio, and D. Eck. A distance model for rhythms. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008c.

Lucas Parra and Uday Jain. Approximate kalman filtering for the harmonic plus noise model. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 75–78, 2001. NOTE: Probabilistic description of the harmonic plus noise model that permits the development of a Kalman filter that tracks pitch.

Geoffroy Peeters and Xavier Rodet. Automatically selecting signal descriptors for sound classification. In *ICMC 2002*, Goteborg, Sweden, september 2002. NOTE: Design of the CUIDADO classifier based on discriminant analysis and mutual information.

S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19: 380–393, 1997.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, February 1989.

Lawrence R. Rabiner and Ronald W. Schafer. *Digital Processing of Speech Signals*. Prentice Hall, 1978.

C. Raphael and J. Stoddard. Harmonic analysis with probabilistic graphical models. *Computer Music Journal*, 28(3):45–52, 2004.

Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing*, 10(1-3):19–41, Januray 2000.

D. Rizo, J.M. Inesta, and P.J. Ponce de León. Tree model of symbolic music for tonality guessing. In *Proc. of the Int. Conf. on Artificial Intelligence and Applications, AIA 2006*, pages 299–304, Innsbruck, Austria, 2006. IASTED, Acta Press. ISBN 0-88986-404-7.

Douglas L. T. Rohde. Methods for binary multidimensional scaling. *Neural Comput.*, 14(5):1195–1232, 2002. ISSN 0899-7667. doi: http://dx.doi.org/10.1162/089976602753633457.

D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2-3):117–149, 1996.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2nd edition edition, 2002.

S. Sadie, editor. *The New Grove Dictionary of Music and Musicians*. St. Martin's Press, 1980.

Lawrence K. Saul, Daniel D. Lee, Charles L. Isbell, and Yann LeCun. Real time voice processing with audiovisual feedback: toward autonomous agents with perfect pitch. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1205–1212, Cambridge, MA, 2003. MIT Press. NOTE: Real time front end for detecting voiced speech and estimating its fundamental frequency.

E. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustical Society of America*, 103(1):588–601, 1998. URL http://web.media.mit.edu/~eds/beat.pdf.

Eric D. Scheirer. *Music-Listening Systems*. PhD thesis, Massachusetts Institute of Technology (MIT), June 2000.

E. Schellenberg. Simplifying the implication-realization model of musical expectancy. *Music Perception*, 14(3):295–318, 1997.

Paul Schmeling. *Berklee Music Theory Book 1*. Berklee Press, pap/com edition edition, 2005.

J. F. Schouten. The perception of timbre. In *Reports of the 6th International Congress on Acoustics*, pages 35–44, Tokyo, 1968.

A. Sheh and D. P. Ellis. Chord segmentation and recognition using EM-trained Hidden Markov Models. In *Proceedings of the 4th ISMIR*, pages 183–189, Baltimore, Maryland, October 2003.

Chuck Sher, editor. *The New Real Book*, volume 1-3. Sher Music Co., 1988.

Malcolm Slaney. Mixtures of probability experts for audio retrieval and indexing. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, Lausanne, Switzerland, August 2002. NOTE: Two different mixture-of-probability-expert models are trained to learn the association between acoustic queries and the corresponding semantic explanation, and visa versa.

Malcolm Slaney and Richard F. Lyon. A perceptual pitch detector. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 357–360, 1990. NOTE: Pitch detector based on Licklider's "Duplex Theory" of pitch perception.

Hagen Soltau, Tanja Schultz, Martin Westphal, and Alex Waibel. Recognition of music types. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 1998. NOTE: Explicit Time Modelling with Neural Networks.

Andrew D. Sterian. *Model-Based Segmentation of Time-Frequency Images for Musical Transcription*. PhD thesis, University of Michigan, 1999.

B. Thom. Predicting chords in jazz: the good, the bad, and the ugly. In *IJCAI-95, Music and AI Workshop*, Montreal, Canada, 1995.

D. M. Titterington, A. F. M. Smith, and U. E. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley, 1985.

George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5), July 2002. NOTE: Results comparable to those obtain with humans are obtained using different features as inputs to classifiers.

George Tzanetakis, Georg Essl, and Perry Cook. Automatic musical genre classification of audio signals. In *Proceedings of the International Symposium on*

Music Information Retrieval (ISMIR), Bloomington, Indiana, 2001. NOTE: Describing a set of features for representing texture, instrumentation, rythmic structure and strength. Evaluation of these features for classification purposes.

George Tzanetakis, Georg Essl, and Perry Cook. Human perception and computer extraction of beat strength. In *Proceedings of the Conference on Digital Audio Effects (DAFX)*, Hamburg, Germany, September 2002. Description of automatic beat strength measures.

V. Valimaki, J. Huopaniemi, Karjaleinen, and Z. Janosy. Physical modeling of plucked string instruments with application to real-time sound synthesis. *J. Audio Eng. Society*, 44(5):331–353, 1996.

V. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

P. Vassilakis. Chords as spectra, harmony as timbre. *J. Acoust. Soc. Am.*, 106 (4/2):2286, 1999. (paper presented at the 138th meeting of the Acoustical Society of America).

Paul Joseph Walmsley. *Signal Separation of Musical Instruments*. PhD thesis, Department of Engineering, University of Cambridge, September 2000. NOTE: Simulation methods for musical signal decomposition and transcription.

Patrick Zanon and Gerhard Widmer. Recognition of famous pianists using machine learning algorithms: First experimental results. Technical Report OEFAI-TR-2003-01, ÖEFAI, 2003. NOTE: A very interesting application of musical genre recognition.

J. Ziv and A. Lempel. Compression of individual sequences via variable rate coding. *IEEE Trans. Inf. Theory*, 24(5):530–536, 1978.

Jean-François Paiement

941, rue de Monaco
Mont Saint-Hilaire, QC
Canada, J3H 4N4

Home : 1 (450) 464-3963
paiement@idiap.ch
www.idiap.ch/~paiement

EDUCATION

Doctorate: Ph. D. in Computer Science, Ecole Polytechnique de Lausanne ; Switzerland — 2004-2008
Probabilistic Models for Music Analysis and Generation. Supervised by Samy Bengio.

Master: M. Sc. in Computer Science, Université de Montréal ; Canada — 2002-2003
Out-of-Sample Extensions For Dimensionality Reduction Algorithms. Supervised by Yoshua Bengio. CGPA 4.2/4.3.

Diploma: B. Sc. in Mathematics and Computer Science, Université de Montréal ; Canada — 1999-2002
Graduated with Honors. CGPA 3.8/4.3. Last year in Université Joseph-Fourier, Grenoble, France, in an exchange program.

PROFESSIONAL EXPERIENCE

Research Intern, Google ; USA — August 2007- March 2008
Image Ranking using the PAMIR model. Supervised by Samy Bengio.

Research Assistant, IDIAP Research Institute ; Switzerland — 2004-2007
Probabilistic Models for Music Analysis and Generation. Supervised by Samy Bengio.

Research Assistant, LISA, Université de Montréal ; Canada — 2001-2003
Neural Networks for Data-Mining in a Marketing Application, with Bell Canada.

Teaching Assistant, Université de Montréal ; Canada — 2000-2001
Advanced Programming in C.

Summer Intern, RALI, Université de Montréal ; Canada — 2000
Music Generation Using Functional Programming. Supervised by Guy Lapalme.

Musical Director, Ozias-Leduc High School Big Band ; Canada — 1996-1999

Private Piano Teacher ; Canada — 1993-2003

AWARDS AND SCHOLARSHIPS

Doctoral Research Scholarship, FQRNT ; Canada — 2004-2007

Masters Research Scholarship, FQRNT ; Canada — 2002-2003

Exchange Program Scholarship, Agence Universitaire de la Francophonie ; France — 2001-2002
This scholarship was given to only one student in North America.

Undergraduate Research Award, NSERC ; Canada — 2000

Grant for Specialized Music Sound Recording, Canada Council for the Arts — 2000-2001

Most Outstanding Student Award, Saint-Laurent College ; Canada — 1997

SKILLS

Programming in C++, Matlab, C, Objective-C, Java, and Haskell.

Mathematical skills in machine learning, information retrieval, statistics, probability theory, analysis, and algebra.

Strong **pedagogical** experience in both collective and individual teaching**.**

Piano mastering.

PUBLICATIONS

1. **J.-F. Paiement**, Y. Grandvalet, S. Bengio, and D. Eck. A Distance Model for Rhythms. *Proceedings of the 25th International Conference on Machine Learning (ICML)*, to appear, 2008.
2. **J.-F. Paiement**, Y. Grandvalet, S. Bengio, and D. Eck. A Generative Model for Rhythms. *Proceedings of the Music, Brain, and Cognition Workshop, Advances in Neural Information Processing Systems (NIPS)*, 2007.
3. **J.-F. Paiement**, D. Eck, and S. Bengio. Probabilistic Melodic Harmonization. *Proceedings of the 19th Canadian Conference on Artificial Intelligence*, 2006.
4. **J.-F. Paiement**, D. Eck, and S. Bengio. A Probabilistic Model for Chord Progressions. *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2005.
5. **J.-F. Paiement**, D. Eck, S. Bengio, and D. Barber. A Graphical Model for Chord Progressions Embedded in a Psychoacoustic Space. *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005.
6. **J.-F. Paiement,** Master thesis (in French): Généralisation d'algorithmes de réduction de dimension. Université de Montréal, 2004.
7. Y. Bengio, O. Delalleau, N. Le Roux, **J-F. Paiement**, P. Vincent, and M. Ouimet. Learning Eigenfunctions Links Spectral Embedding and Kernel PCA. *Neural Computation*,16(10):2197-2219, 2004.
8. Y. Bengio, **J.-F. Paiement**, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. *Advances in Neural Information Processing Systems 16 (NIPS)*, 2004.
9. Y. Bengio, P. Vincent, **J.-F. Paiement**, O. Delalleau, M. Ouimet, and N. Le Roux. Spectral Clustering and Kernel PCA are Learning Eigenfunctions. Technical Report 1239, Département d'informatique et recherche opérationnelle, Université de Montréal, 2003.
10. Y. Bengio, **J.-F. Paiement**, P. Vincent. Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. Technical Report 1238, Département d'informatique et recherche opérationnelle, Université de Montréal, 2003.
11. Y. Bengio, P. Vincent, and **J.-F. Paiement**. Learning Eigenfunctions of Similarity: Linking Spectral Clustering and Kernel PCA. Technical Report 1232, Département d'informatique et recherche opérationnelle, Université de Montréal, 2003.