Generating Music with Graphical Models

Loic L'Heveder

MSc by Research in Pattern Analysis and Neural Networks

ASTON UNIVERSITY

September 2000

Acknowledgements

I would like to thank David Barber for his great support, wise advice and enlightened discussions. I also thank the NCRG staff, Kevin Murphy and all the people who helped me complete this work on music.

ASTON UNIVERSITY

Generating Music with Graphical Models

Loic L'Heveder

MSc by Research in Pattern Analysis and Neural Networks, 2000

Thesis Summary

Music can be seen as a set of pieces of sound, a set of instrument combinations or a set of notes. Modeling a song in one of these ways provides the possibility to study what we are going to play with respect to the previous times. Because of this discrete splitting of a piece of music, we thought of using graphical models. This thesis is dealing with the use of Markov Models and family in order to model music. We were interested in building a model complex enough to catch the local information of a measure and the more global information of a set of measures.

Keywords: Music, HMM, CMM, DBN

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Music has two facets: its source, musical scores, and its result, sound. Analyzing, synthesizing and reproducing Music is not a new interest in the neural networks field. Two approaches of a piece of music mainly appeared: the first one considers Music through its result, a sound signal, and brings to use waveform models [12], the second one tends to consider Music through its source, as a succession of discrete events [6][8]. In this thesis we use the second approach, working on the source.

When listening to a piece of music, we hear a sequence of sounds, a sequence of different frequencies. Looking at a musical score, all instruments are described distinctly, each of them defined by an individual score, showing what and when to play. The central question driving this thesis is, can we model the structure in a musical score?

The central aims of this thesis are twofold:

(1) To model musical structure in order to be able to generate music. That is, we want a model which shows *"creativity"*.

(2) To make a model with a "help to creation" function. With even the most complex and high-performance stand alone model, creating music automatically is difficult. Rather than replacing the composer, we can use the computer to help the composer: the composer has an idea of the main melody and/or rhythm, and can ask the model to find a completion of this idea.

1.1 Choosing the model

A musical score represents music in a discrete space. The basic assumption that we make in choosing a model, is that a score can be described by a sequence of transitions in this discrete space, from one step to the next, describing a trajectory through the discrete space. In order to create novel musical sequences, we work within a probabilistic framework. Defining *"what we play"* wwp_{now} and given wwp_{before}, these transitions can be modelled by the probability of playing wwp_{now} a certain way:

$$P(wwp_{now}|wwp_{before})$$

Our approach is based on the assumption that a music piece is a combination of elements from a basis. A stochastic model could then produce a new sequence of combinations.

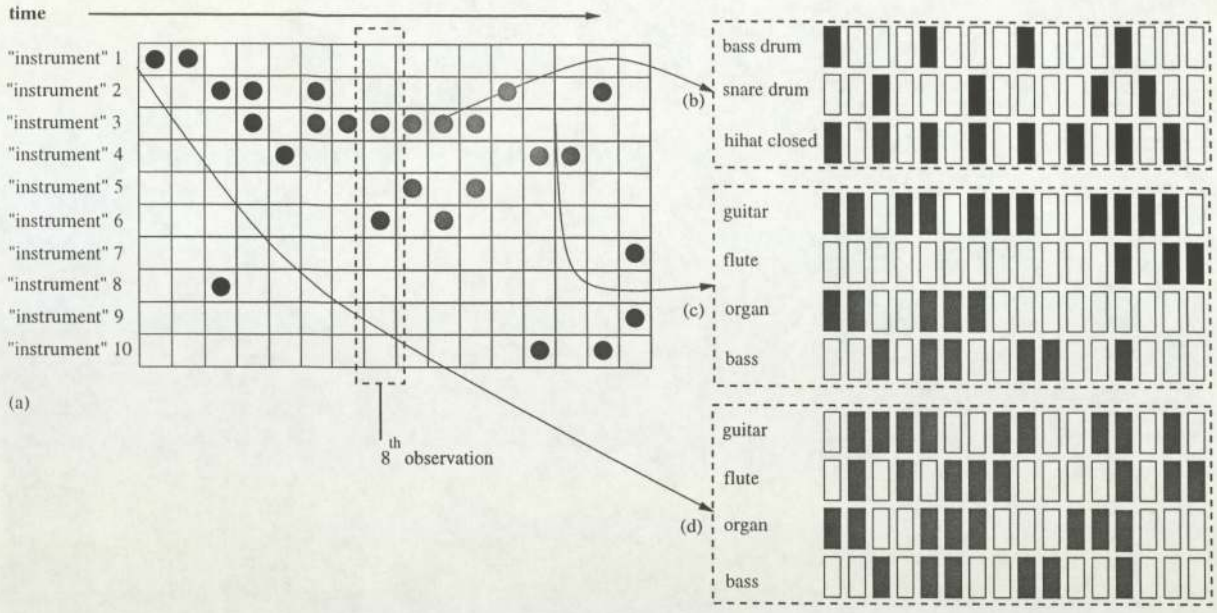


Figure 1.1: *instruments combination* representation. Dividing the song in time slices, at time t, a combination of "instruments" is played. Each "instrument" can be a standard instrument sound, or itself a fixed piece of sound played by a set pattern of other instruments. The combination of the small pieces played at t will be the state of our model at time t.

In short, our basic assumption is that music can be modelled by an underlying stochastic, dynamic Markov process. That is, choosing an appropriate time scale, the future is not dependent on the past given the present.

1.2 Music Representation

A central issue intimately connected with modelling is the representation of the music. Depending on the degree of freedom we want:

- We can see a piece of music as a set of combinations of instruments, each combination representing the instruments at a time slice t (fig 1.1). We will call this the *instruments combination* representation.
- We can go deeper in the structure and think of the score as a set of values in $\{$*not played, played way 1, played way 2, ..., played way n*$\}$ per instrument, per time slice, we will call this the *instruments* representation (fig 1.2).

Due to the restrictions of our modelling framework, we will primarily focus on music pieces which are well structured and repetitive in nature. In this sense, pop music is a natural area of interest, and we will therefore concentrate on extracting structure from pop songs.

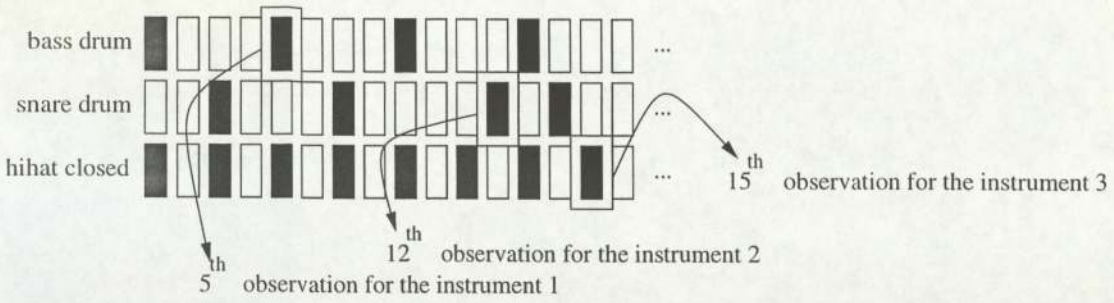


Figure 1.2: *instruments* representation, each instrument is considered separately from the others, following its own underlying process. Our assumption is then that each instrument can be modelled by an underlying stochastic, dynamic process. Music can be modelled by an interaction of these processes across time.

1.3 Rhythm & Melody

Once we have decided how we were going to model a piece of music, we can divide a song in two parts: the rhythm and the melody.

By "rhythm" we mean the sequence generated by the drum, percussions, clave, maracas, timbales, in fact any instrument which has a small number of ways to be played. Thus {*not played, played*} describes the vocabulary of this instrument or group of instruments ({*hit the drum, do not hit the drum*}).

We consider melody to be generated by all the other instruments and voices. Melodic vocabulary is generally much larger. For this reason we will focus mainly on rhythm, since we believe that this can be well modelled by a small vocabulary of a modest number of instruments. A natural starting point for modelling rhythm is basic drum patterns, and extracting structure from a dataset of drumming examples, using our stochastic Markov framework. We begin with simple models, gradually increasing their complexity and exploring the different properties and performances of these models. A general framework in which all the models can be described is Graphical Models. Since these form the backbone of the project, we describe in the following chapter essential details of Graphical Models and means by which they can be used for computation.

Chapter 2

Graphical Models

Graphical Models are a powerful framework for probabilistic data modelling [4]. Such models are a marriage of probability theory and graph theory in which dependencies between variables are expressed graphically. We will use the notation $G(X, L)$ to define a graphical model. $X = \{X_i\}$ is the set of nodes, each of them representing a variable x_i, $L = \{X_i \rightarrow X_j\}$ is the set of directed link between the nodes.

2.1 Bayesian Networks

A Bayesian network, or belief network, is a directed acyclic graph (DAG) in which each node x_i represents a random variable with a probability distribution. The central idea behind graphical models is that every probability distribution admits a kind of factorisation. Given a joint distribution on a set of variables $\{x_i\}_{i=1..n}$, without loss of generality, we may write $p(x_1, ..., x_n) = p(x_1 | x_2, .., x_n)p(x_2, .., x_n)$. We can repeat this process, writing $p(x_2, ..., x_n) = p(x_2 | x_3, .., x_n)p(x_3, .., x_n)$ and so on. Each factor is a conditional distribution of a single variable, conditional on another. These factors define a *conditional probability table* (CPT). In practice, typically, not all variables will directly influence each other. Considering $\{pa(i)\}_{i=1..n}$, the parents of the variables $\{x_i\}$, the graphical structure and the set of parameters specify a joint distribution over the random variables,

$$p(x_1, .., x_n) = \prod_i p(x_i | pa(i))$$

When x_i has no parents, the table is a single vector representing the distribution of this variable over its domain.

> ***Example***: let us consider the network $G(X, L)$ in fig 2.1, $X = \{A, ..., G\}$, each variable is binary, $L = \{a \rightarrow d,\ b \rightarrow d,\ c \rightarrow f,\ d \rightarrow f,\ d \rightarrow g,$ $e \rightarrow g\}$. This directed acyclic graph represents a joint distribution on X, $p(a, b, c, d, e, f, g)$. Using the independency structure of the variables induced by L, we can rewrite the joint probability distribution (JPD) of the network as follows:
>
> $$p(x) = p(a)p(b)p(c)p(e)p(d | a, b)p(f | c, d)p(g | d, e) \qquad (2.1)$$

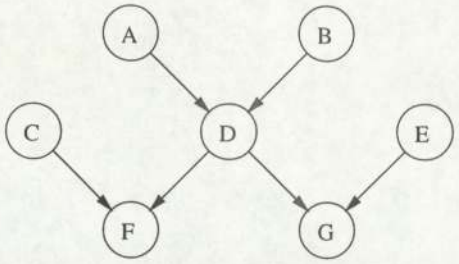


Figure 2.1: simple DAG

$a \setminus b$	"0"	"1"
"0"	2/3	0
"1"	2/3	1

(a)

$a \setminus b$	"0"	"1"
"0"	1/3	1
"1"	1/3	0

(b)

Table 2.1: Conditional probability table for the node D from the DAG in fig 2.1 (a) $p(d = \text{"0"}|a, b)$ (b) $p(d = \text{"1"}|a, b)$. All other nodes similarly require a specific conditional probability table for the joint distribution to be fully specified.

Thus the graphical structure, fig 2.1, defines the independency relationships between the variables. However, the exact form of the conditional probability tables also need to be specified, for example as given by table 2.1.

2.1.1 Inference

The task of calculating conditional (marginal) distributions from the full joint distribution is called inference. Such tasks typically occur in the case that certain nodes are clamped to particular values. These values constitute *"evidence"*, and inference with evidence mathematically corresponds to calculating conditional distributions. To give a flavour of inference we consider a simple example.

One morning, Oliver goes into his garden and realizes that the grass is wet (O). Is it due to rain (R) or has he forgotten to turn off the sprinkler (S)? Looking to the garden of his neighbour, Claudia, he remarks that it is also wet. He thinks therefore that it is more likely it rained. Let us see how this evidence that Claudia's grass is wet can affect the explanation of why Oliver's grass is wet.

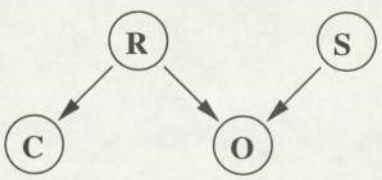


Figure 2.2: Belief network structure for the "wet grass" example, modelling the relationships between the variables C (Claudia's grass is wet), O (Oliver's grass is wet), R (it has rained) and S (the sprinkler was left on).

We can model the situation with a DAG as in fig 2.2. Having defined the structure, we still need to define the specific values associated with each of the CPTs. The prior probabilities for R and S are given by:
$p(R = yes) = 0.2$ and $p(R = no) = 0.8$, written as $p(R) = (0.2, 0.8)$. $p(S) = (0.1, 0.9)$. Let us define the other CPTs as $p(C = y|R = y) = 1$. $p(C = y|R = n) = 0.2$, as sometimes Claudia also leaves her own sprinkler on. $p(O = y|R = y, S) = 1$. $p(O = y|R = n, S = y) = 0.9$, as there is a small chance that the grass is not noticeably wet even if the sprinkler were left on. $p(O = y|R = n, S = n) = 0$.
The joint probability is given by

$$p(R, S, O, C) = p(O|R, S)p(C|R)p(R)p(S) \tag{2.2}$$

The prior belief that the sprinkler is responsible is $p(S = y) = 0.1$. Let us calculate the marginal $p(S = y|O = y)$ using Bayes rule:

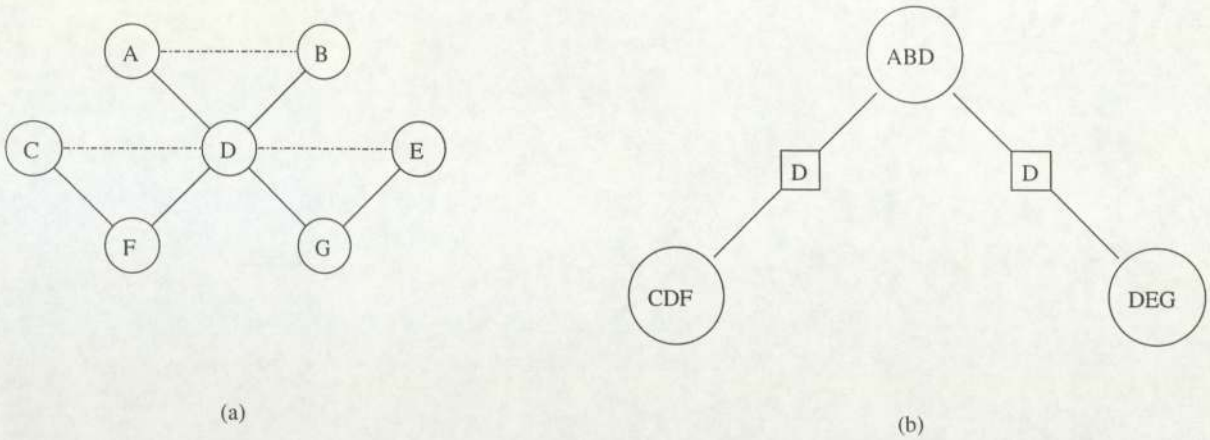
$$\begin{aligned}
p(S = y|O = y) &= \frac{p(S = y, O = y)}{p(O = y)} && (2.3)\\
&= \frac{\sum_{C,R} p(O = y, C, R, S = y)}{\sum_{C,R,S} p(O = y, C, R, S)} && (2.4)\\
&= \frac{\sum_{C,R} p(C|R)p(O = y|R, S = y)p(R)p(S = y)}{\sum_{C,R,S} p(C|R)p(O = y|R, S)p(R)p(S)} && (2.5)\\
&= \frac{\sum_{R} p(O = y|R, S = y)p(R)p(S = y)}{\sum_{R,S} p(O = y|R, S)p(R)p(S)} && (2.6)\\
&= \frac{0.9 * 0.8 * 0.1 + 1 * 0.2 * 0.1}{0.9 * 0.8 * 0.1 + 1 * 0.2 * 0.1 + 0 * 0.8 * 0.9 + 1 * 0.2 * 0.9} && (2.7)\\
&= \frac{0.092}{0.272} = 0.3382 && (2.8)
\end{aligned}$$

This shows that the fact that the grass is wet *increases* the belief, beyond the prior belief that the sprinkler is on.
Now, let us calculate $p(S = y|O = y, C = y)$ using Bayes rule again:

$$\begin{aligned}
p(S = y|O = y, C = y) &= \frac{p(S = y, O = y, C = y)}{p(O = y, C = y)} && (2.9)\\
&= \frac{\sum_{R} p(O = y, C = y, R, S = y))}{\sum_{R,S} p(O = y, C = y, R, S)} && (2.10)\\
&= \frac{\sum_{R} p(C = y|R)p(O = y|R, S = y)p(R)p(S = y)}{\sum_{R,S} p(C = y|R)p(O = y|R, S)p(R)p(S)} && (2.11)\\
&= \frac{\sum_{R} p(O = y|R, S = y)p(R)p(S = y)}{\sum_{R,S} p(O = y|R, S)p(R)p(S)} && (2.12)
\end{aligned}$$

Substituting in the numbers, we get

$$p(S = y|O = y, C = y) = \frac{0.0344}{0.2144} = 0.1604 \tag{2.13}$$



(a) (b)

Figure 2.3: (a) moralized graph associated to the DAG in fig 2.1 (b) Junction Tree associated with (a). The square nodes are "separators", indicating variables shared between neighbouring cluster nodes.

So the probability that the sprinkler is on, given Claudia's grass and Oliver's grass are wet, is *lower* than given that only Oliver's grass is wet.

Although straightforward on a small network like the example above, when the number of variables (nodes) is large, inference is often computationally non-trivial. In a directed acyclic graph, however, we can use the independency structure of the probabilistic model to attempt to find efficient inference algorithms. A general algorithm that achieves this is called the Junction Tree algorithm. This will be described in brief in subsection 2.1.1, more details of the algorithm can be found in [4].

Junction Tree Algorithm

The main goal in building a Junction Tree is to modify the form of a DAG $G(X, L)$ to an equivalent structure in order to make the inference calculations easier. The Junction Tree represents the same joint probability distribution as the original graph and the structural dependencies of G are preserved. We will not go into any details of how calculations are performed on the Junction Tree, and refer the interested reader to a standard textbook such as [4]. However, of importance here is the computational effort required to perform calculations with a model. This is determined by the clique[1] sizes in the Junction Tree, and we therefore explain briefly how these are determined. The construction of a Junction Tree is performed in several steps:

- **moralizing**: remove the direction of the links in order to obtain an undirected graph (the direction information will remain in the CPDs).
- **triangulation**: every loop of length four or more contains at least one chord. An example is displayed in fig 2.4.

The computation time on the Junction Tree scales exponentially with the clique size, so that graphs with even modest clique sizes rapidly become intractable. A crucial issue

[1] A subset of nodes S of a graph G is said to be complete if there are links between every pair of nodes in S. S is called a clique if S is not a subset of another complete set. In fig 2.3 (a), the cliques are {C, D, F}, {A, B, D} and {D, E, G}.

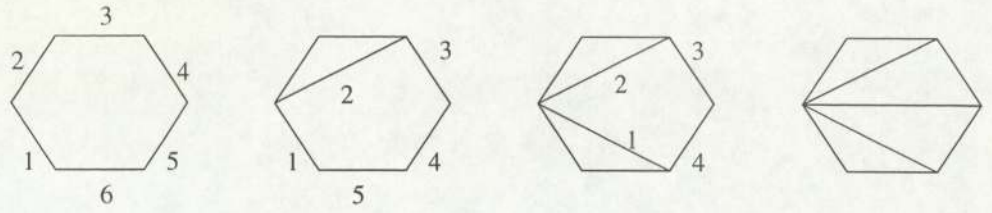


Figure 2.4: Triangulation process applied on a loop of length 6. From left to right, we see the recursive application of the definition.

O	R	S
y	y	y
n	n	y
y	y	n
y	y	n
n	n	n
y	n	y
n	n	n

Table 2.2: Data for the sprinkler example. Each row represents an observation made on Oliver's grass, the rain and the sprinkler at a time t

in designing a graphical model, therefore, is the clique sizes of the resulting Junction Tree.

2.1.2 Learning in Graphical Models

The goal of learning graphical models [7] is to find the values of the parameters of each CPT in order to maximize the likelihood of the training data. Considering N cases in the training set $D = \{X^1, .., X^N\}$ and a model $G(X, L)$ with M nodes, the normalized log-likelihood of D is a sum of terms, one for each node:

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} \log p(x_j | pa(x_j), X^i) \tag{2.14}$$

The log likelihood function contains a term for each node in the graph. Hence, the maximization can be carried out on each node independently.

If all the nodes are observable, for example with Markov models (section 3.1), and follow a multinomial distribution, the learning process amounts to counting the number of occurrences of a value for a node x_j with respect to the values of the parents $pa(x_j)$. Let us give a simple example. Considering fig 2.2, suppose we have the training observation data presented in table 2.2, representing 7 independent observations of the variables O, R and S. This information is sufficient to determine the CPT $p(O|R, S)$. We can find the maximum likelihood estimate of the CPT of node O by counting the number of times Oliver's grass is wet when it has rained and the sprinkler was on, $N(O = y, R = y, S = y)$, the number of times Oliver's grass was wet when it has rained and the sprinkler was off, $N(O = y, R = y, S = n)$, and so on. Then we can

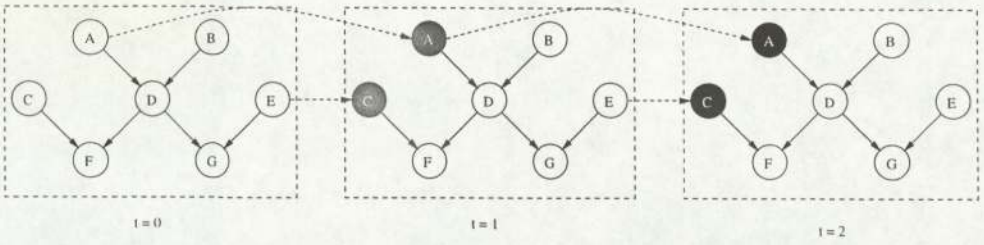


Figure 2.5: Dynamic Bayesian Network associated with DAG in fig 2.1, the three first time slices are shown, assuming the inter-slice dependencies $a \rightarrow a$ and $e \rightarrow c$

write

$$p(O = o|R = r, S = s) = \frac{N(O = o, R = r, S = s)}{N(R = r, S = s)}, \quad o, r, s \in [y, n] \tag{2.15}$$

where $N(R = r, S = s) = N(O = y, R = r, S = s) + N(O = n, R = r, S = s$. We obtain the probabilities $p(O = y|R = y, S) = 1$, $p(O = y|R = n, S = n) = 0$, $p(O = y|R = n, S = y) = 0.5$, $p(O = n|R = y, S) = 0$, $p(O = n|R = n, S = 0) = 1$ and $p(O = n|R = n, S = y) = 0.5$.

In this example all the nodes were observable. This is not always the case, for example with hidden Markov models in section 3.4. When some of the nodes are hidden, we can use the *Expectation Maximization* algorithm where we compute the expected values of all the nodes using an inference algorithm, then consider them as observed values in order to maximize the likelihood in the example [4].

2.2 Dynamic Bayesian Network

A dynamic model may be defined as a sequence of submodels each representing the state of a dynamic system at a particular point or interval in time. Such a time instance is referred to as a time slice. A Dynamic Bayesian Network (DBN) consists of a series of subnetworks, in our case structurally identical Bayesian networks, interconnected by temporal relations (links). It models the stochastic evolution of a set of variables over time. This set, at a certain point t in time, is called the t^{th} slice of the DBN. We will consider the belief network in fig 2.1. A DBN is therefore defined by the intra slice connections, and inter slice connections, specifying how to connect two temporally adjacent slices [13]. As we are translating $G(X, L)$ over time, the structure of each slice is the same and can be defined as $G_t(X_t, L)$, as depicted in fig 2.5. Let us explain briefly what the relation between the slices is and how the Junction Tree for this graph is formed.

In our example here, we will work with only two time slices. The first one represents the initial states of our model. The second one contains the structure which is going to be unrolled along time. The moralized, triangulated graph in fig 2.3(a) of the unrolled graph contains three complete subsets giving rise to three cliques of size 3. In fig 2.6, we see now the triangulation across time necessitates adding several inter links. Thus, the triangulated graph is modified, and there is now a clique of size 4 and three cliques of size 3. Since computation time increases exponentially with clique size [1, 2], this increase from the size 3 cliques in the unrolled graph is bad news. Some care needs

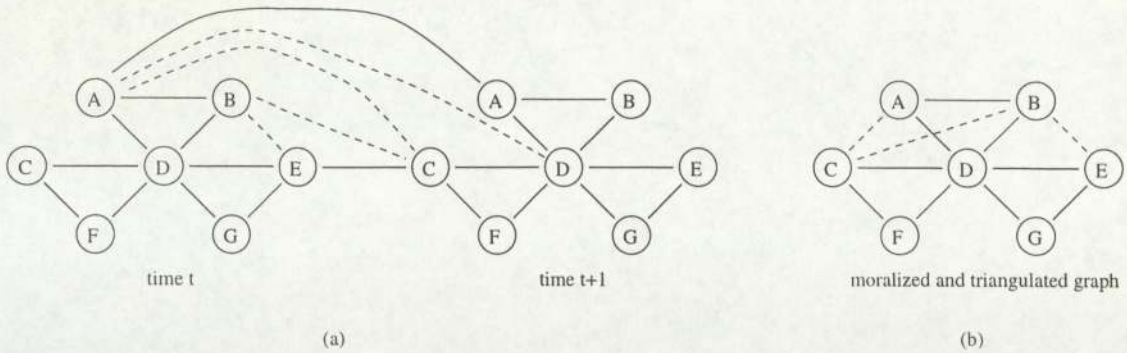


Figure 2.6: In (a), we see the junction tree composed of a slice at time t and the slice at $t + 1$. Building a Junction Tree over two time slice, the triangulation process lead to the addition of more inter links, and therefore the Junction Tree in one slice (b) is modified

to be taken, therefore, that clique sizes do not get out of hand in designing Dynamic Bayesian Networks.

Chapter 3

Basic Models

Markov models, a special case of Dynamic Bayesian Networks, have been used for almost 30 years, and their rich mathematical structure and has been applied to several important applications including speech recognition [11] and genetic sequence analysis [9]. Using Markov models for music is based on the idea that there exists an underlying stochastic dynamic process generating the music. The complexity of this process corresponds to different levels of complexity and structures of the Markov models. In this chapter, we describe initially the simplest of these basic Markov type models. This will lead us to consider another interesting type of graphical model, tree-structured belief networks.

3.1 Markov Models

Studying a set of observable events, ordered in time, an intuitive model would be to associate a node per event (*states*) and work out the probability to have an event S_i after S_{now}.

Let us consider a system where, regularly in time, we can observe a certain state S_i out of a set of $\tilde{N}$ states. We denote the system being in state S_i at time t by $S_i(t)$, and a probabilistic representation of a Markov process is given by

$$p(S_i(t) \mid S_j(t-1),\ S_k(t-2),\ ...) \tag{3.1}$$

That is, given that the system is in state S_j at time $t-1$, state S_k at time $t-2$, etc, we associate a probability of the state at the current time being S_i. The full specification of the distribution for all the states at time t, $S(t) = \{S_i(t)\}_i = 1..\tilde{N}$, dependent on previous states is written

$$p(S(t) \mid S(t-1),\ S(t-2),\ ...) \tag{3.2}$$

A simplifying assumption can be made by considering the current state to be dependent on only the previous n-steps, a so-called n^{th}-order Markov chain. We will consider the simplest one, the first order process which describes distributions of the form

$$\begin{aligned} p(S(t),\ S(t-1),\ S(t-2),\ S(t-3),\ ...) = \\ p(S(t) \mid S(t-1))p(S(t-1) \mid S(t-2))p(S(t-2) \mid S(t-3))... \end{aligned} \tag{3.3}$$

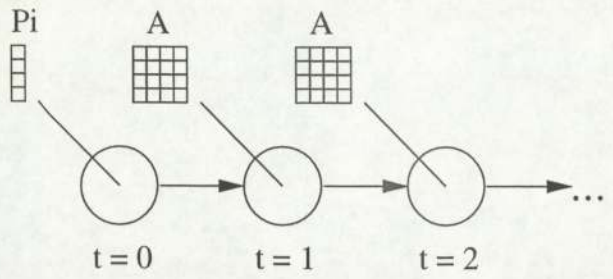


Figure 3.1: The graphical model representation of a Markov process is a one node chain, the node taking its value in $\{S_1, S_2, ..., S_{\tilde{N}}\}$

Assuming stationarity of the process, the chain transitions are fully described by the $\tilde{N}$ by $\tilde{N}$ transition matrix A

$$A = \{a_{ij}\} = \{p(S_j(t) \mid S_i(t-1))\} \tag{3.4}$$

with

$$a_{ij} \geq 0 \ \textit{and} \ \sum_{j=1}^{\tilde{N}} a_{ij} = 1 \tag{3.5}$$

We also need to define the probability distribution over the states at time $t = 1$

$$\Pi = \{\pi_i\} = \{p(S_i(t=1))\} \tag{3.6}$$

The graphical model description of a Markov model is a straightforward chain structure, as depicted in fig 3.1.

We can use this model to learn the structure of our data. The likelihood of the sequence of (musical) states, $S_{i_1}, ..., S_{i_n}$, is given by

$$\begin{aligned} p(data) &= p(S_{i_1}, S_{i_2}, ..., S_{i_n}) && (3.7)\\ &= p(S_{i_1}) \cdot p(S_{i_2}|S_{i_1}) \cdot \ ... \cdot p(S_{i_n}|S_{i_{n-1}}) && (3.8)\\ &= \pi_{i_1} \cdot a_{i_1 i_2} \cdot ... \cdot a_{i_{n-1} i_n} && (3.9) \end{aligned}$$

The parameters Π and A can then be adjusted to maximize the likelihood of the observed sequence. The optimal setting of the transition parameters under maximum likelihood corresponds to the intuitive notion of simply counting the frequency of state to state transitions.

3.2 Learning The Structure of a Simple Song

In figure 3.2, we show three examples of simple songs. Along the vertical axis are represented the "instruments" (actually small segment of sounds), and the horizontal axis represents time. These examples are given in file *mm_example1.wav*, *mm_example2.wav* and *mm_example3.wav*.

We trained a separate Markov model for each song. A state in the Markov model will correspond to a unique "instrument" combination, of which there are 8 in example 1. The Markov model then captures the transitions between these 8 states.

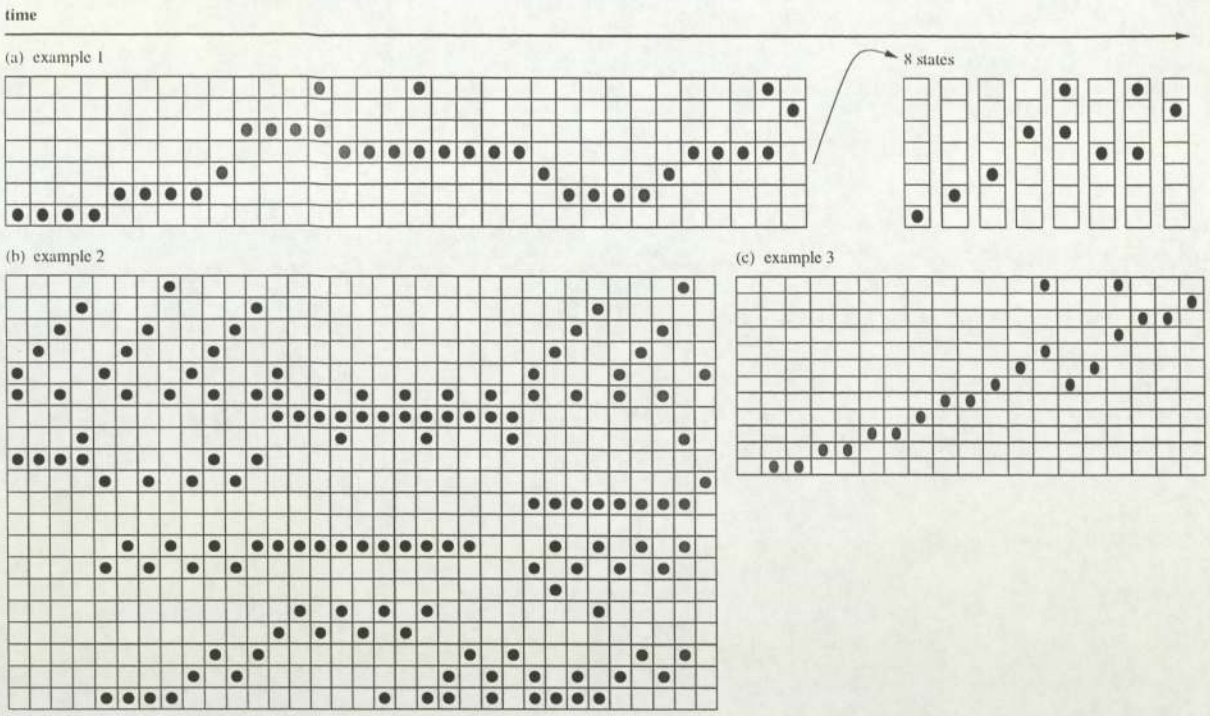


Figure 3.2: Matrix representation of three songs we used to train our Markov models, the vertical axis represents an "instrument" and the horizontal axis, time. A separate Markov model was trained on each song.

3.2.1 First order Markov Models

Finding the transition matrix A and prior distribution matrix Π is readily achieved by counting the number of transitions from state i to state j, and normalizing such that

$$\sum_j a_{ij} = 1, \ \forall i. \tag{3.10}$$

The initial state sets the prior probability $\pi_{i_1} = 1$ and $\pi_{i_k} = 0$, $k = 2..n$. We can then sample from this Markov model to produce novel sequences. In sampling, we select an initial state according to the prior distribution Π, say S_j. The next state is then sampled from the distribution defined by the j^{th} column of the transition matrix.

Training a separate Markov model on each of the songs in fig 3.2, we then sampled from the trained model sequences of same length. We show such samples in fig 3.3. Comparing the sequence of states in the original song and the corresponding sample, we see that the sequences are reproduced with local similarity to the training data, however with a lack of global structure. These examples are given in files *mm_1_sample1.wav*, *mm_1_sample2.wav*, *mm_1_sample31.wav* and *mm_1_sample32.wav*. For example 3, we have presented two samples from the trained Markov model. Note that the samples, whilst similar to the local transition structure of the training, are both different. Additionally, sample (d) highlights an inherent weakness in the first order approach. There is a transition from one state to two equally likely states, such that the model can be "stuck" in one of these states for too long.

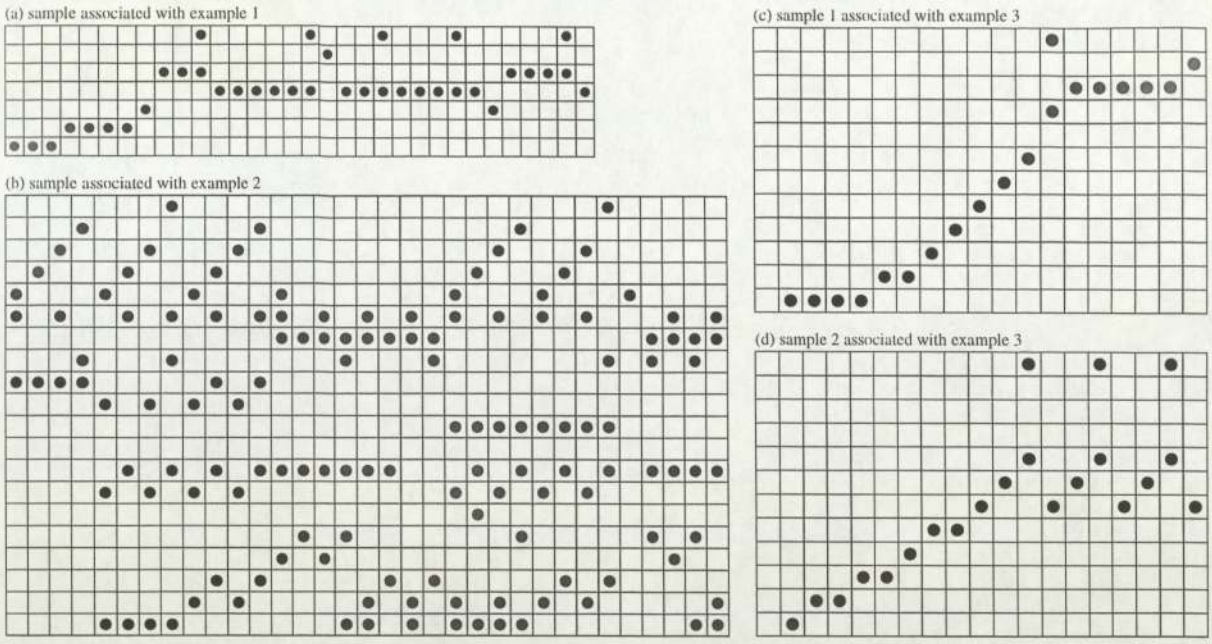


Figure 3.3: Using a first order Markov model, we train it on song shown in fig 3.2(a), then we sample from the distribution represented by the trained model, a simple sample from which is given in (a). (b) is sampled from a model trained on example fig 3.2(b). (c) and (d) are two samples from a model trained on example fig 3.2(c).

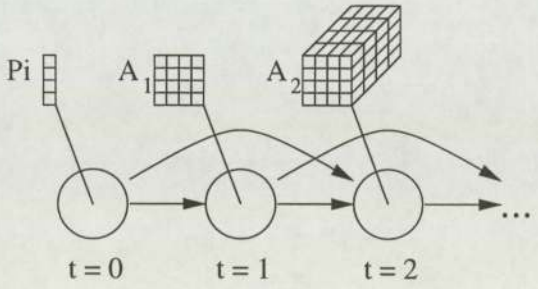


Figure 3.4: 2^{nd} order Markov process.

3.2.2 Higher order Markov models

Although the first order Markov model displays some *"creativity"*, the assumption that the present state depends only on the previous one is too strong to allow the model to catch more global structure. One approach to improve this is increase the order of model. Choosing a second order Markov model, fig 3.4, we trained it on the same songs as in fig 3.2 and obtained the samples in fig 3.5, given in *mm_2_sample.wav*.

We see that, in addition to reproducing the local sequence structure, the samples are closer to the original songs in terms of global structure. This confirms that by increasing the complexity of our model, we increase the flexibility of the structure and its ability to capture the temporal dimension in the sequence of outputs. However the models are too complex or the data representation not appropriate, since the creativity is almost nonexistent.

Table 3.1 shows the log likelihood of the first and second order models on the three examples, and clearly shows that the second order fits more closely to the data, even

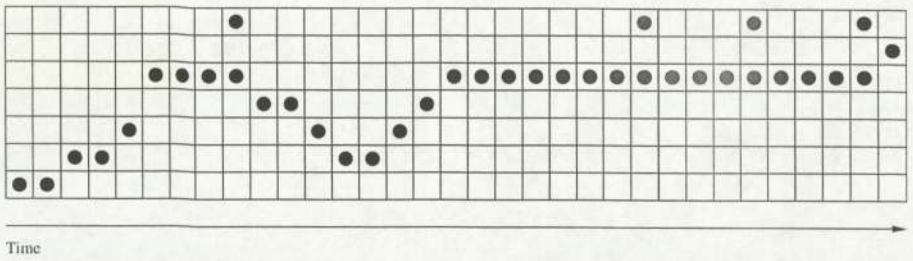


Figure 3.5: This sequence is sampled from a 2^{nd} order model trained on example fig 3.2(a), for the two others the model overfitted the data as shown in table 3.1 with a zero log likelihood.

	1^{st} order	2^{nd} order
example 1	-21.3579	-11.0904
example 2	-3.2958	0
example 3	-8.3178	0

Table 3.1: Log likelihoods on the three examples in fig 3.2 with a first and second order Markov model.

perfectly for some of the data (example 3). This arises from the representation of the music in which most of the transitions appear only once, making our stochastic process essentially deterministic.

Conclusion

Using a first order Markov model captures the local structure of the music. However, the music generated by this approach does not sound structured enough. Using a higher order Markov Model improves this situation. By increasing the order of the Markov process, we more faithfully reproduce the original song. However, this is at the expense of limiting the creative possibilities of the model.

How can we achieve both aims of improving the global structure of the model, without simply reproducing the training sequence exactly? One way to do this is to use higher order models with a limited complexity - hidden Markov models (HMMs). These will be described in some detail in section 3.4.

3.3 Textbook Drum Sequences

A limitation associated with the training data used in the Markov models of section 3.1 is that different songs use a different set of instruments, so that we used a separate model for each song. This limits rather severely the amount of training data available for the model. For much of the rest of the thesis, we use instead a set of textbook rock music drum sequences[1]. This consist of a set of 63 sequences, each being 64 beats in length, with a vocabulary of instruments being bass drum, snare drum, hihat open and hihat closed.

[1] *"Rudimentary patterns for the modern drummer"*, Cusatis - , *"Rhythmic patterns for the modern drummer"*, Cusatis

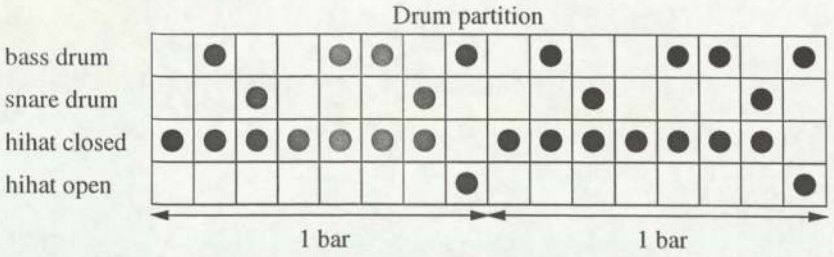


Figure 3.6: matrix representation of a drum partition. Each column represents one input for our model, using equation 3.11 in order to give a unique value to this combination of instruments.

A short, 16 beat segment of one sequence is presented in fig 3.6 where, at each beat, or time, an instrument is either played, or not played.

The number of states is then 2^4, each state representing a binary vector $(I_{bass\ drum}, I_{snare\ drum}, I_{hihat\ closed}, I_{hihat\ open})$. For example, the state representing playing only snare drum and hihat opened at time t is

$$(I^{(t)}_{bass\ drum}, I^{(t)}_{snare\ drum}, I^{(t)}_{hihat\ closed}, I^{(t)}_{hihat\ open}) = (0,1,0,1) = 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5 \tag{3.11}$$

In fig 3.7, we present a subset of the 63 drum partitions we used to train our hidden Markov model. A suitable bar length is based on 16 time steps. Some of the examples have an original tempo half that of the others. We use then an empty state for those sequences to ensure that all the examples are on the same time scale. For the moment, each instrument is played *on beat*, which means in one of the eight parts of the bar. We will see in chapter 4 how to deal with *off beat* notes by extending the vocabulary of an instrument. To hear a selection of these elemental drum sequences, play the files *elem_example1.wav* and *elem_example2.wav*. Note that we divided the data set in two sets, a training set composed of 50 drum examples and a test set with the remaining 13.

3.4 Hidden Markov Models

For a state space of dimension N, there are N^2 parameters to learn in the transition matrix A of a first order Markov process. Many of the transitions will occur infrequently in the training data, making the estimation of the parameters of the transition matrix difficult. Hidden Markov models are an extension of Markov models, in which a hidden, unobserved process generates the observed sequence of data, see fig 3.8.

A hidden Markov model is defined by

- A set of hidden states $S=\{S_1,...,S_N\}$ generally all reachable from any S_i
- The state transition probability distribution $A = \{a_{ij}\}$ such that

$$a_{ij} = p(S_j^{(t+1)} \mid S_i^{(t)}), \ \forall\ (i,\ j) \in [1..N]^2 \tag{3.12}$$

- The initial state distribution $\Pi = \{\pi_i\}$ such that

$$\pi_i = p(S_i^{(1)}), \ \forall\ i \in [1..N] \tag{3.13}$$

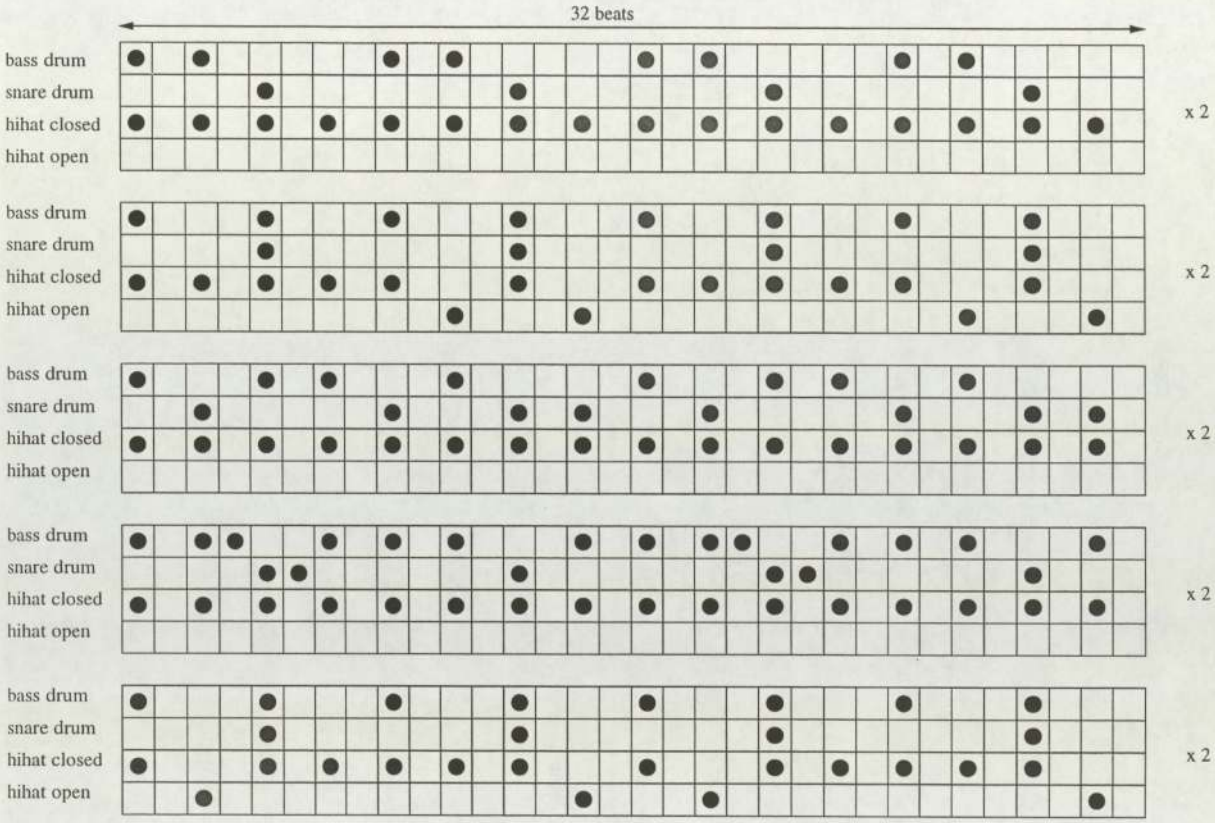


Figure 3.7: Matrix representation of a selection of the drum partitions used as data set, the notation is the one used in fig 3.6. The vertical axes of each matrix represents the instruments, bass drum in the first row, snare drum in the second one, hihat closed in the third and hihat opened in the last one. The horizontal axis represents time, from left to right. The black ball stands for an instrument to be played.

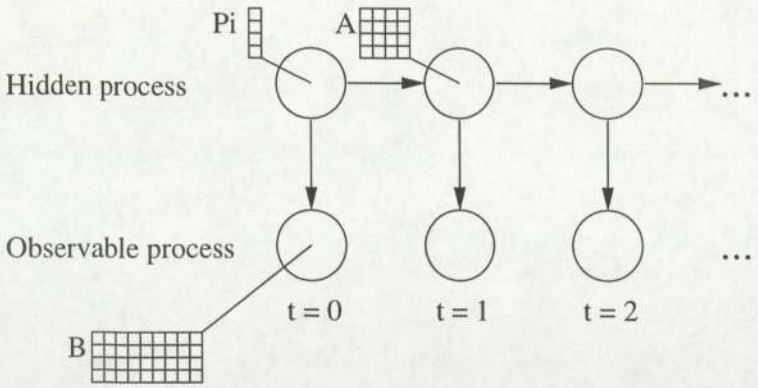


Figure 3.8: Dynamic representation of a first order hidden Markov model

The observation symbol probability distribution is modeled by

- An observation space with vocabulary, $V=\{v_k\}_{k=1..M}$
- A probability distribution over these symbols from each state i, $B_i=\{b_i(k)\}$, such that

$$\forall\,(k)\ \in\ [1..M],\ b_i(k)\ =\ p(v_k^{(t)}\mid S_i^{(t)}) \tag{3.14}$$

This defines a joint distribution on the hidden and visible (observable) variables as

$$p(S^{(1)},V^{(1)},S^{(2)},V^{(2)},...,S^{(n)},V^{(n)}) = \\ p(S(1))p(V^{(1)}|S^{(1)})p(S^{(2)}|S^{(1)})p(V^{(2)}|S^{(2)})...p(S^{(n)}|S^{(n-1)})p(V^{(n)}|S^{(n)}) \tag{3.15}$$

Once both stochastic processes have been defined, the task of learning an observation sequence $V = \{v^{(1)},...,v^{(n)}\}$ lies in maximizing the likelihood of the observed sequence $p(V \mid A,B,\Pi)$. To do so, we used the standard Expectation-Maximization algorithm (EM) [11].

3.4.1 Higher Order Hidden Markov Models

In our experiments concerning standard Markov models, we found that basing the transition on only the previous time step is too restrictive, and that higher order Markov models are more able to capture accurately the dynamics. In a similar way, we can define higher order hidden Markov models. As shown in fig 3.9, a higher order model can be considered as a dynamic bayesian network. Looking at slice 2 in fig 3.9, we see that we need to define two transition matrices A_4 and A_5, and an inter slice transition matrix A_3. A standard, stationary HMM would constrain all these matrices to be equal, although here we will allow them to take different values. These matrices are, however, fixed across slices, so that $A_6 = A_3$, $A_7 = A_4$ and $A_8 = A_5$, etc.

3.4.2 Results and Interpretation

The hidden structure (states and transitions matrices) models the output by a lower dimension underlying structure. Using a lower number of states than the number

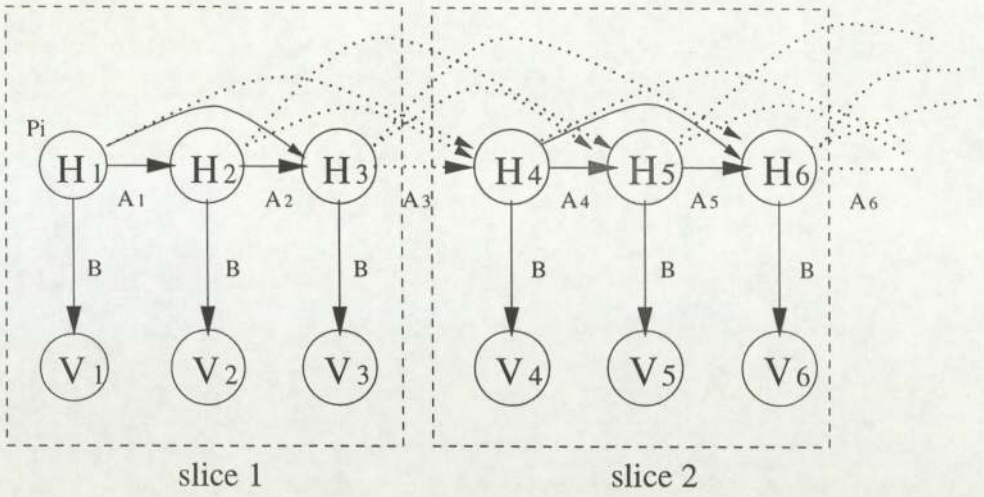


Figure 3.9: A third order hidden Markov model can be expressed as a Dynamic Bayesian Network using slices of width 3, and defining appropriate inter and intra slice connections.

of vocabulary elements, the training algorithm will tend to use all hidden states in producing the observed sequences. However, this lower dimensional representation encourages creativity, since the confusion matrix B tends to be less certain in mapping the hidden state to an observed state. That is, for a hidden Markov model with a small number of hidden states, the hidden state does not identify a particular observation state with any certainty but identifies a distribution on the observation vocabulary. We trained several HMMs on our data, with different order and number of hidden states in order to find a suitable model complexity.

Up to the 2^{nd} order, the results coincide with our expectations: the higher the order, the better the model fits the data, training and test set, see figures 3.10 and 3.11. Also, when we increase the number of hidden states, this tends to increase the log likelihood without overfitting. That is, it extracts from the training set enough information to model the unseen data. However, the 4^{th} order results are an exception, with a tendency to reduce the likelihood on the training data as the number of hidden states is increased beyond 6. This is likely to indicate that the EM algorithm struggles to avoid local minima in the complex error surface.

The choice of the order is important because it relates to the depth of the information retrieval process. We have chosen order 4 because most of our examples are based on a tempo multiple of 4, in addition to good performance in our experiments. Moreover we used a time scale equal to a 16^{th} of a bar, *i.e* to create our matrix representation of a piece of music, we cut each bar in 16 time slices. Examining figures 3.10 and 3.11, we additionally chose 4 hidden states for the fourth order model. In fig 3.12 we show a sequence generated from a 4^{th} order HMM within a model of slice width 4, the whole sequence is given in *hmm_4_4_sample1.wav*.

At this point, the model we are using provides a good approximation of the local structure, but fails to capture more global structure. One approach would be to use a small number of long range connections, see fig 3.13. As an alternative, we consider in the following section tree-structured networks.

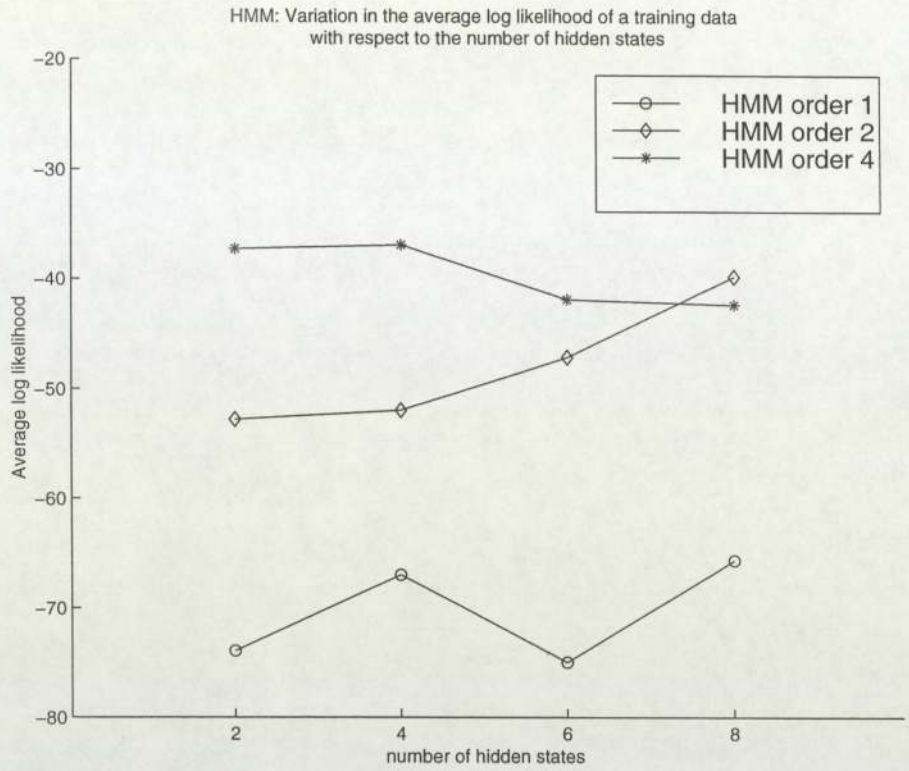


Figure 3.10: We compare here the average log likelihood of the training set of some hidden Markov models of different order with the number of hidden states.

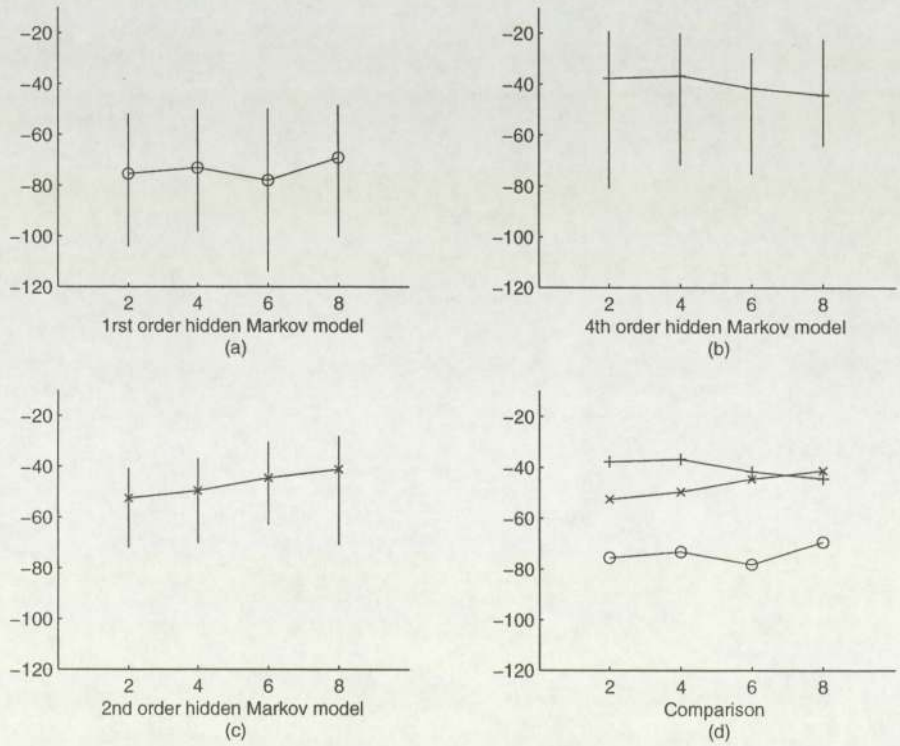


Figure 3.11: Comparison of the average log likelihoods of the test set data for higher order hidden Markov model depending on the number of hidden states on the horizontal axis. We represent also the lowest and highest log likelihood of the 13 test examples.

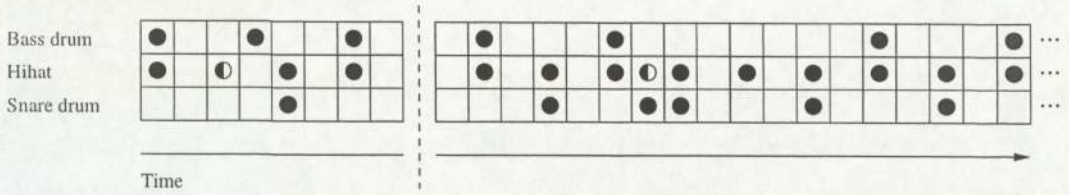


Figure 3.12: Sequence generated from a 4^{th} order, 4 hidden states Markov model

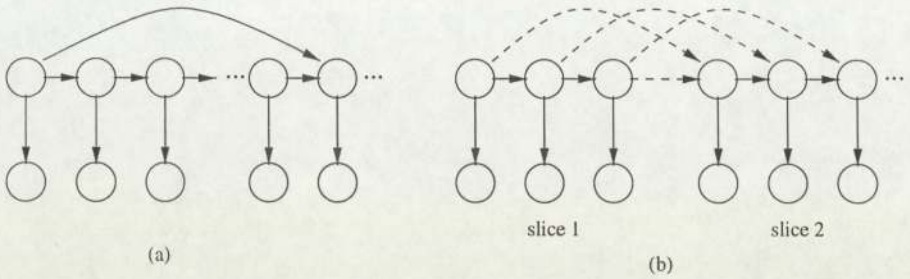


Figure 3.13: Representation of a long range connections structure. A state at time t may be more dependent on a state many time steps before. For example, the first note of each bar may well depend directly on the first note of the previous bar. Figure (a) shows the structure while figure (b) presents the associated Dynamic Bayesian Network.

3.5 Tree-structured Markov Models

The general model of a tree-structured belief network is illustrated in fig 3.14 [14]. The observed data **O** are assumed to be the output of an underlying and unobserved process **S**. **S** is a multi-level tree, S^0 is the top node. $\boldsymbol{S}^L$ represents the lowest level, composed of the variables $\{S_i^L\}$ directly connected with the observed data. Each node S_i^k has its children in $\boldsymbol{S}^{k+1}$. This structure is a graphical model in which each node is associated with a CPT

$$p(S_i^k|Pa(S_i^k))\,,\ Pa(S_i^k)\ \in\ \mathbf{S^{k-1}}$$

The topology of a tree-structured model is dependent on the size of the data. Even

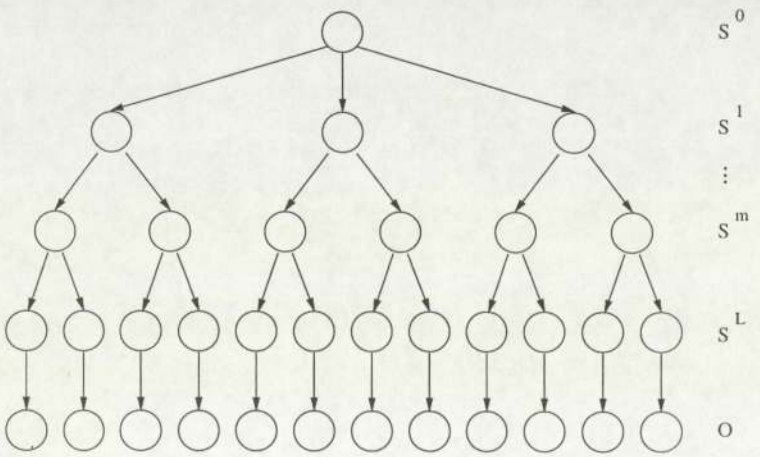


Figure 3.14: a $L+1$ tree-structured model

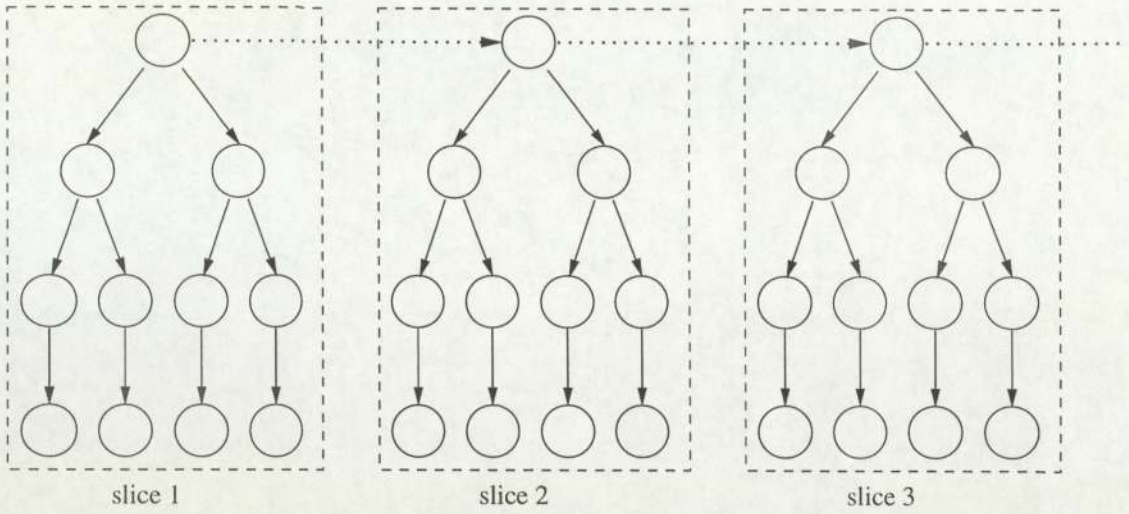


Figure 3.15: dynamic bayesian network based on a tree-structured belief network

if our data set contains examples of the same length, we want to keep the possibility of working with sequences of different lengths. We are going to use a dynamic tree-structured graph, fig 3.15, so that the temporal information will be local in each slice, represented by the probability table of the top node S^0. We impose a Markov process built on that set of nodes

$$p(S_t^0 \mid S_{t-1}^0, ..., S_1^0) = p(S_t^0 \mid S_{t-1}^0) \tag{3.16}$$

ensuring continuity over the time slices.

We show in fig 3.17 the log likelihood, sum of $\log(p(D_i|model))$ on all the drum partitions, of some tree-structured models, for both training and test data. Each model has its structure described in fig 3.16. The results show that a simpler structure like tree-structured belief networks can model our data with a reasonable success, providing a log likelihood of both data set and test set higher than the HMM of order less than 4. It appears that a multi-level structure can extract information from the set of observations contained in a slice. A single first order Markov process based on the top level node is enough to recreate the local information of the set of observations. Two examples sampled from the structure in fig 3.16(a) are given in file *ts_*.wav*[2].

However, over the test set, the results are even worse than with a 2^{nd} order HMM with 8 hidden states. Tree-structured belief networks seem to be a good model to carry global information but do not capture well the local information. An advantage of such a model using Dynamic Bayesian Networks is that its associated Junction Tree is composed of cliques of size 2 and 3, therefore the learning process is performed faster than for more complex models. Initially designed to be applied on the whole sequence of observations, the dynamic higher level structure does not seem to carry local information across time. We will see in section 4.1 how to use an adaptation of tree-structured models in order to add to our model a higher level structure to capture global information from our data.

[2]Although generated from the same model, the two examples have been recorded with two different set of instruments. In the second example, *ts_1_12_sample.wav*, a bongo replaces the snare drum, electronic drum sounds replace the other classic drum sounds.

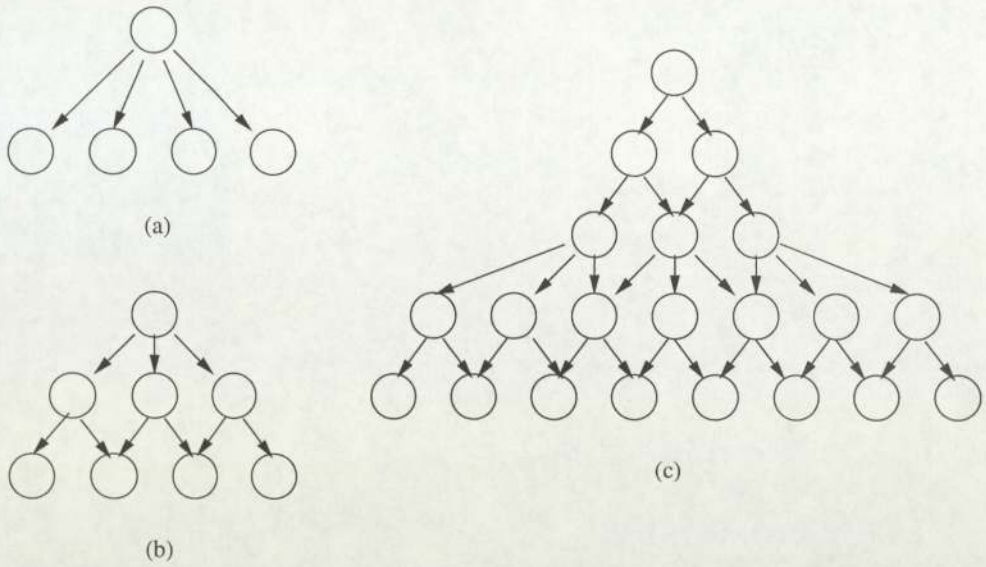


Figure 3.16: This figure shows 3 different tree-structured models we trained on the data set. The nodes at the bottom of the structures are the observation nodes.

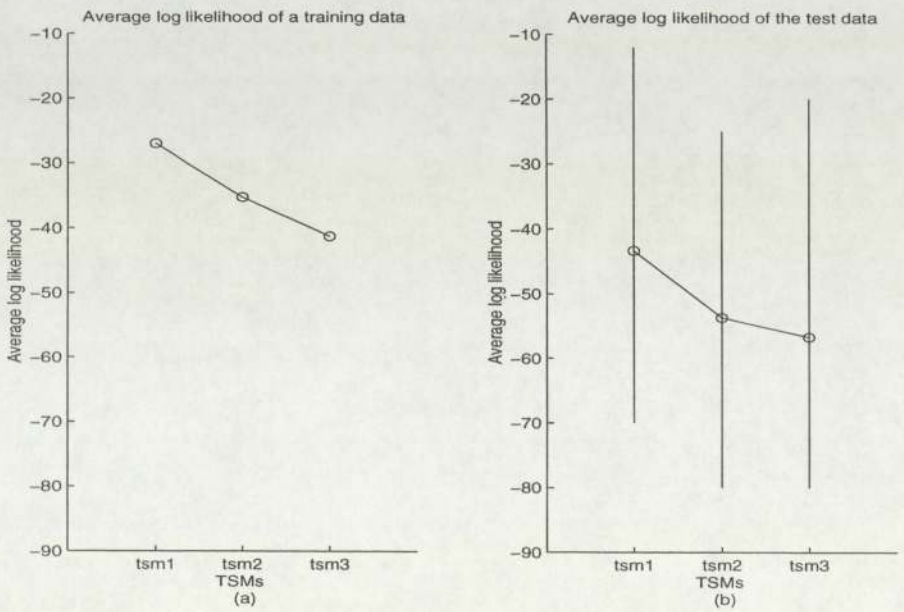


Figure 3.17: This figure shows log likelihood of three tree-structured models, their structures are described in fig 3.16. For the model *tsm*(a) we used 12 hidden states. *Tsm*(b) and *tsm*(c) have been trained with respectively 8 and 6 hidden states due to computation limitation, the clique sizes should remain reasonable.

3.6 Conclusion

Using hidden Markov models and an adaptation of tree-structured belief networks, we can capture quite well local rhythmical structure. All the samples generated from our higher order hidden Markov models, fig 3.12 and files *hmm_*.wav*, reproduced parts of the training examples, generating also novel sequences. One important feature that they respect is the tempo, giving consistency to the rhythm. Nevertheless, we are concerned about two points. Firstly, there is still a lack in the global structure: the samples do not maintain stability in the rhythm during several bars. Increasing the order of the model could improve the results but at the expense of the creativity. Secondly the *instruments combination* representation for music is too still constraining. It does not allow us to exploit the inter dependencies between instruments and the fact that we should perhaps weight differently the influence of an instrument on itself and on the others.

The next chapter describes the use of a *supervising* structure in order to catch the global information and create a backbone for the model, in addition to allowing a more flexible representation of the data.

Chapter 4

Coupled Models

Until now, we used the *instruments combination* music representation to create our data set. Perhaps, a more natural conception of music is to see each instrument with its own structure, *e.g.* we know how to play the bass drum alone as much as the bass drum and the snare drum together. A piece of music then results from the time interaction between several instruments. Thus, we use in this chapter the *instruments* representation in fig 1.2, where each instrument is associated with a Markov process (fig 4.1). This has the interesting advantage of allowing us to clamp certain instruments in particular states and calculate the inference on the others.

To create piece of music the instruments have to play in harmony. We are now interested in coupling these processes in order that they interact to create a suitable output.

This representation has the advantage to reduce the number of hidden states of our model. If there are P processes with N_p states each, an HMM will require N_p^P states as a coupled model will use PN_p states. Therefore, we will attribute one process to each "physical" instrument. In fig 4.3, we show how we extended the vocabulary of hihat in order to group hihat closed and open as one instrument.

One way to extend the hidden Markov model framework in order to couple processes evolving in parallel is to couple them at the output, see fig 4.2(a), and is generally known as a *factorial hidden Markov model* (FHMM). The signal of each process is overlaid with the others in a single output signal. In the FHMM structure we consider, each triplet of nodes $\{X_{bass\,drum}, X_{snare\,drum}, X_{hihat}\}$ is linked to shared output nodes. The second way is to couple the process themselves, each of them having its own output signal, fig 4.2(b). This represents a division of the system, music, in different components with complementary information.

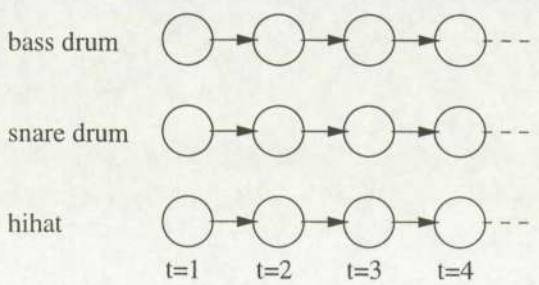


Figure 4.1: Each instrument is modelled by its own Markov process

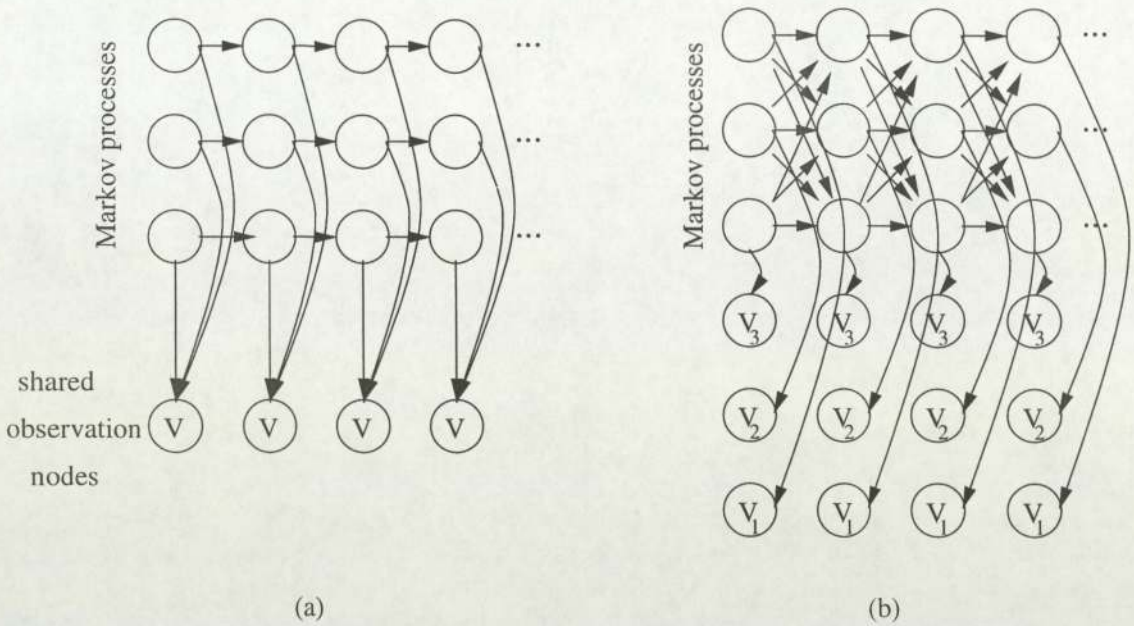


Figure 4.2: We want to couple the instrument processes such that they interact to produce the output signal. Figure (a) shows the three Markov chains coupled at the output, forming a factorial hidden Markov model. Figure (b) shows the three Markov chains coupled across time, forming a *coupled hidden Markov model* .

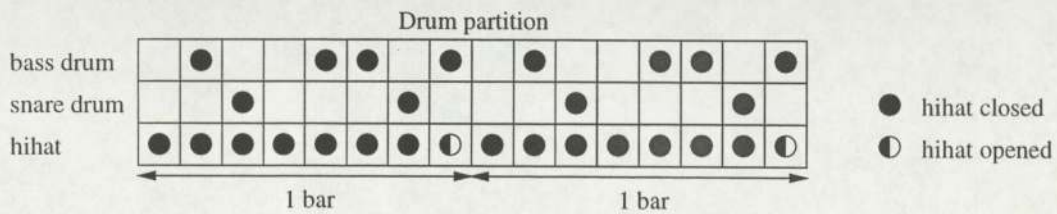


Figure 4.3: Matrix representation of a drum partition when extending the vocabulary of the hihat. Contrary to fig 3.6, each instrument is associated with its own input, giving three input per column. We extended the vocabulary of the *hihat closed* instrument in order to play the hihat opened as different way to play the same instrument. Therefore the size of the vocabulary of the hihat is 3, 2 for the other instruments and the number of states is $2^2 * 3$.

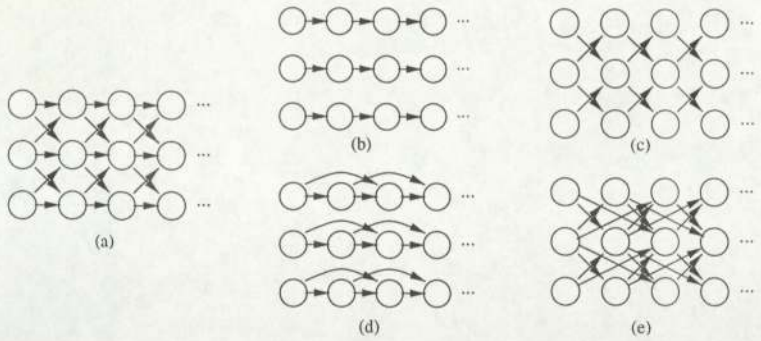


Figure 4.4: This figure shows in (a) the graphical representation of a coupled Markov model. To simplify the notation, the links between the nodes of the top row of (a) and those of the bottom row are not drawn. The figures (b) and (d) show the links considered to remain to the time dimension for, respectively, 1^{st} and 2^{nd} time dimension order coupled Markov model. The figures (c) and (e) show the links considered to remain to the instrument dimension for, respectively, 1^{st} and 2^{nd} instrument dimension order coupled Markov model

Interaction between the instruments can, however, be assumed to be possibly both causal, temporal influence, and also asymmetric: at time t the snare drum $sd^{(t)}$ might be more influenced by the bass drum at $t-1$, $bd^{(t-1)}$, and at time $t+1$ it might be more influenced by the hihat at t than $bd^{(t)}$. Thus we consider that the interprocess influences are across time, the links between two processes will have to bridge time slices. Therefore, we choose the graphical representation in fig 4.2(b), the so-called *coupled Markov model* (CMM) [3].

In our data set, the hihat is played in a very simple way and it will not necessitate a hidden process as complex as the one for bass drum. For the moment, the vocabulary of each instrument is small (<5), therefore we consider each sub-process as a Markov model rather than a hidden Markov model. Note that the size of the state space has been reduced from $2^4 = 16$ for the HMM to $2+2+3 = 7$ for the CMM.

Considering fig 4.4, we define two different dimensions for the model: the time dimension dim_T, along a single instrument across the time slices, and the instrument dimension dim_I, the inter links between different instruments. We wish to study the usefulness of a higher order connection between nodes in both dimensions. In figure 4.4(b), the order of the CMM in both dimension is one: along dim_T an instrument at $t+1$, $I_k^{(t+1)}$, depends only on $I_k^{(t)}$, along dim_I $I_k^{(t+1)}$ it depends only on $\{I_j^{(t)}\}_{j \neq k}$. A coupled Markov model of n^{th} order in dim_T and m^{th} will be referred as a $n * m^{th}$ order CMM. We constrain our model to respect the Markov property, a node from a slice t can not have any children in slice $t+2$ and beyond. Therefore, the width of the time slice of our model is an upper bound of the CMM which is encapsulated in the slice.

We trained the CMM on the same 50 drum sequences as used in chapter 3, experimenting with adjusting the order in both the dim_T and dim_I dimensions. Looking at the figure 4.5 confirms that the higher the order in both dimensions, the better the model. However, the improvement induced by a higher dim_T order is greater than a higher dim_I order. This shows that an instrument is more influenced by itself than by the others even if we must take in account that the data set is not complex and varied enough to really underline a noteworthy comparison of this two dimensions. Although

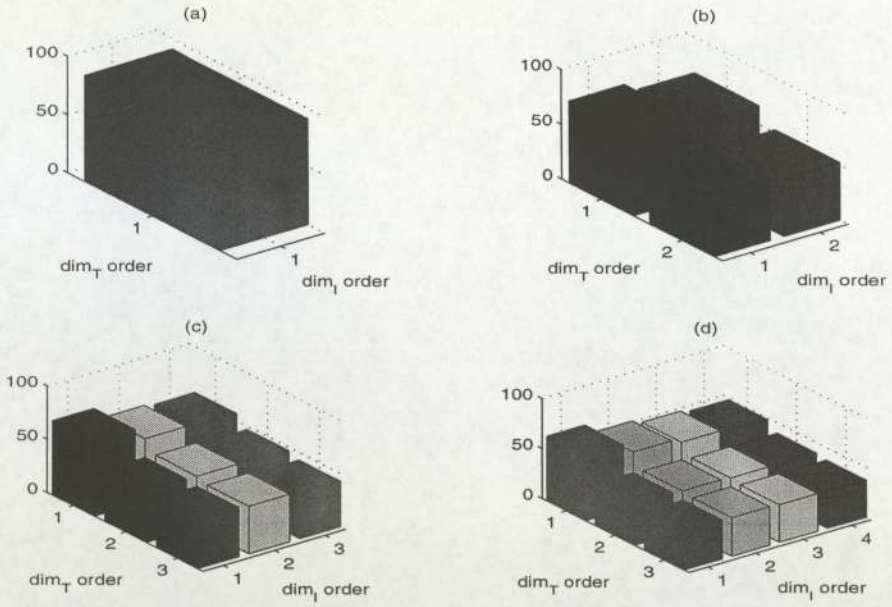


Figure 4.5: Comparison of the average negative log likelihood of a test data for coupled Markov models of different width and different order in dim_T and dim_I. (a) is associated with models of slice width 1, (b) with slice width 2, slice width 3 and 4 respectively for the figures (c) and (d). In the $(\vec{x}, \vec{y}, \vec{z})$, the x axis represents the inter order, the y axis the intra order and the z axis represents the average negative log likelihood of a training data associated with the corresponding model.

increasing both orders increases the log likelihood of the model, it also increases the complexity of the model and, therefore, the computation complexity. Hence, all the models we use in further parts will have a $2^{nd} dim_I$ order and will only be referred by their dim_T dimension. Samples from $2 * 2$ and $4 * 4$ order CMMs are given in files *cmm_*.wav*.

We use figures 4.6 and 4.7 in order to raise two points. First of all, they confirm our belief that a higher model can be used to model the data better without reaching a complexity such that the model overfits the training set, fig 4.7(d). Secondly, we are interested in comparing models of the same order n but encapsulated in a slice structure wider than n. The reason for this is that the wider the slice, the more flexible the model is, since the transition matrices within a slice are not constrained to be equal. Figures 4.7(a) and 4.7(c), we see that, given a Coupled Markov Model with a fixed order, widening the time slice of the model is positively affecting the likelihood of the test data. However, a model with a slice width too large with respect to the order of the CMM is overfitting the data, the flexibility of the structure is too important. For high order models, being in a wider slice does not improve the results, fig 4.7(b), the high order CMM is by itself flexible enough. Therefore, we will work with n^{th} order models within a slice of width n. The results presented in fig 4.7(d) confirm that the higher the order, the better the model is.

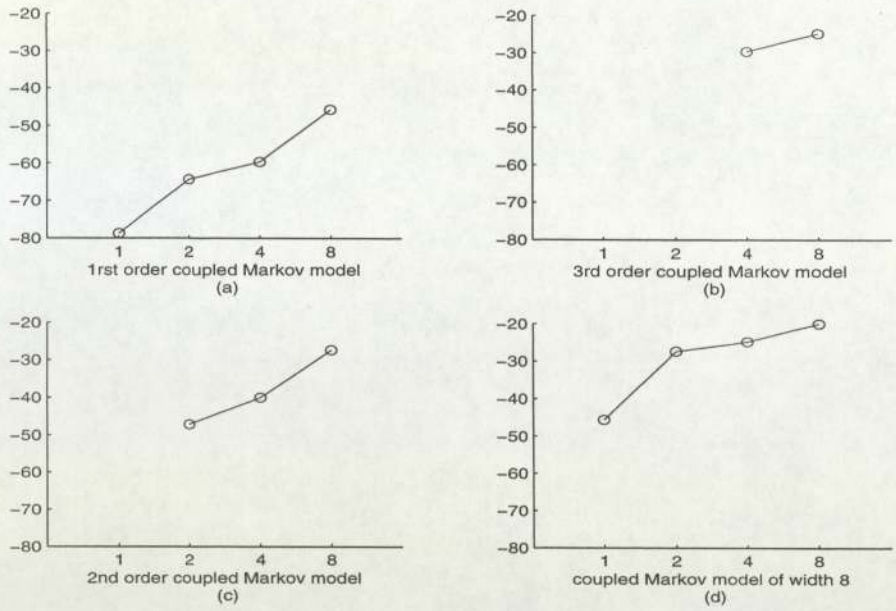


Figure 4.6: This figure concerns the average log likelihood of the training data. The horizontal axis of the three first represents the slice width of the model which encapsulate the CMM. Figure (d) represents the results for a model with time slices of width 8, the horizontal axis show the order of model.

CMM and HMM

In fig 4.8, taking the example of a 4^{th} order structure, we see a modest improvement in modelling using a coupled Markov model rather than a hidden Markov model. Despite capturing the local structure better, to catch global information we need to increase the order of the model, which leads to very large computation times. We prefer to introduce a supervising structure, nodes at a higher level, like in tree-structured models, which will capture the information from a whole slice. Therefore, we introduce some supervised structures in the next section. Samples from a $4 * 2^{th}$ order coupled Markov model, 4^{th} order in time dimension and 2^{nd} in instrument dimension, is partly presented in fig 4.9 and is given in *cmm_4_2_sample3.wav*.

4.1 Supervised structures

The main assumption behind the supervised structure is that music is a multi-level natural stochastic process. At the lowest level the notes follow an underlying process, as modelled by our hidden Markov models or coupled Markov models . However, there also exists some higher-level processes which carry global information, constraining the lower level process. We model these higher processes by including some supervising nodes, nodes from an upper level (level 0) which influence several nodes in the lowest one (level 1).

We consider level 1 as a 4^{th} order coupled Markov model since this gave good performance in our previous experiments. Then we have several different types of supervising structures to work with, as presented in fig 4.10.

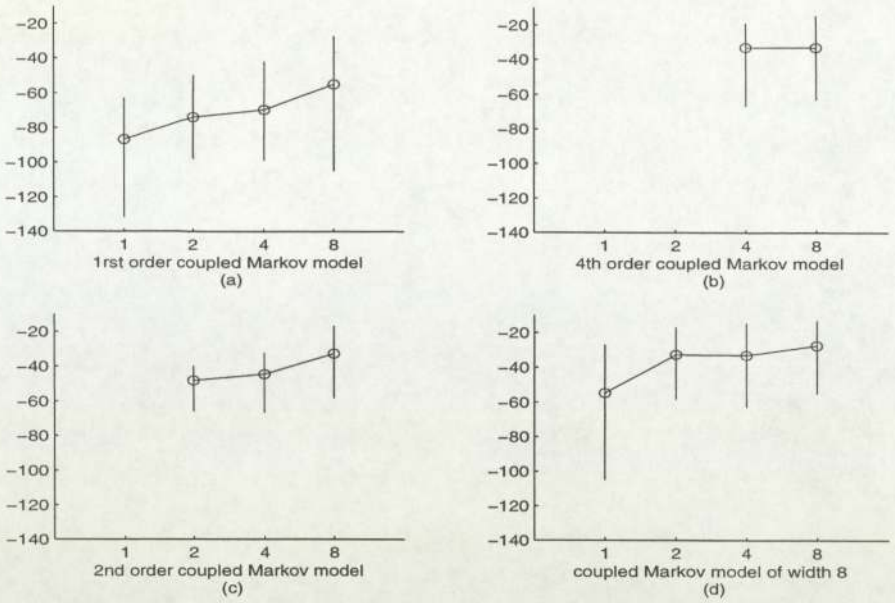


Figure 4.7: This figure shows the evolution of the log likelihoods of the test set data (mean, minimum and maximum). We grouped the model results with respect to the order of the associated model. Then we plot them with respect to the width of their associated models.

One node supervised CMM

The first model in fig 4.10(a) is the simplest and basically attributes a hidden state to one or several combinations of

$bd^{(t)}$	$bd^{(t+1)}$	$bd^{(t+2)}$	$bd^{(t+3)}$
$sd^{(t)}$	$sd^{(t+1)}$	$sd^{(t+2)}$	$sd^{(t+3)}$
$hh^{(t)}$	$hh^{(t+1)}$	$hh^{(t+2)}$	$hh^{(t+3)}$

We then assume that the sequence of such combinations is a stochastic process modelled by a first order Markov chain. The number of hidden states we will associate to the top node will strongly influence the results. A large number of hidden states will give a better likelihood for the training set. The model will overfit the data and reduce the creativity. A small size hidden node will not improve the results compared to a Coupled Markov Model without supervising structure. Therefore, the size of the top node is set to a value between a third and a half of the number of different combinations of instruments, here $2 * 2 * 3 = 12$. In fig 4.12 we see how the addition of this top node with 4 hidden states influences the log likelihood of the training data, capturing with more accuracy the underlying structure as the variance of the log likelihoods of the test data is reduced and their mean is increased.

Time dimension supervised CMM

Fig 4.10(b) shows of another type of supervising structure, each $\{bd^{(t)}, sd^{(t)}, hh^{(t)}\}$ is supervised by a shared node $sv^{(t)}$. We then assume that the evolution of this top node can be modelled by a Markov chain which can be of first or higher order. In fig 4.13 we show a comparison between a 2^{nd} order CMM and its equivalent first and

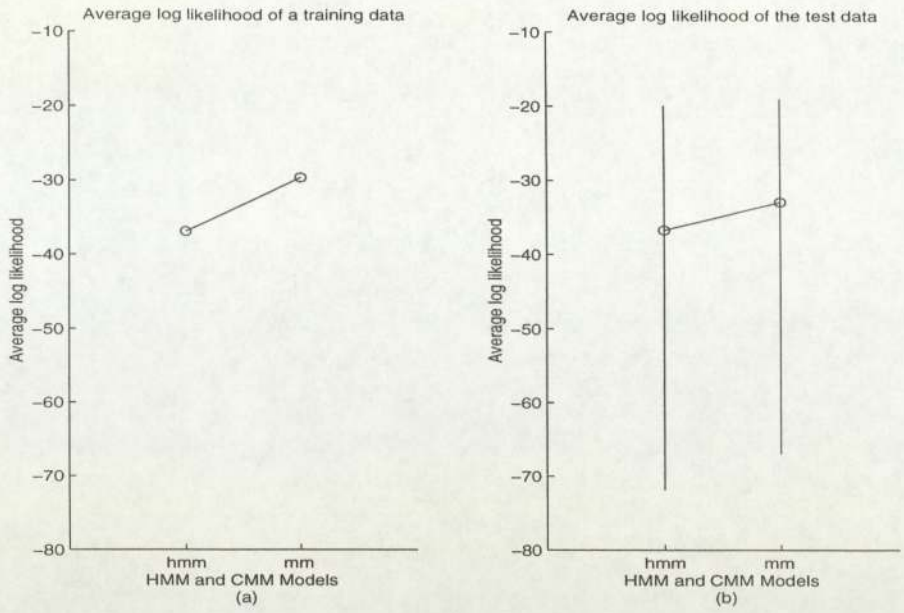


Figure 4.8: Comparison of a 4^{th} order hidden Markov model with a 4^{th} order coupled Markov model. Figure (a) shows the average log likelihood over the 50 elements of the training set. (b) shows the mean of the log likelihoods of the 13 data we kept as test data, together with the worst and best of the 13 test likelihoods for both models

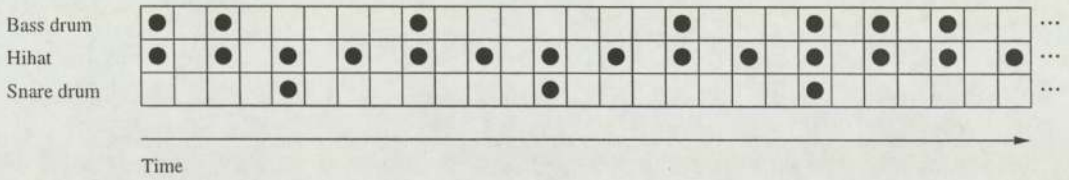


Figure 4.9: Sequence generated from a $4 * 2$ order coupled Markov model

second order *time dimension* supervised models. The size of the supervising nodes is set to 4 for the same reason explained above.

This structure is equivalent to the hidden process in hidden Markov models, except that the observations are interacting. The supervising node ensures a sort of lockstep evolution of the combination of the three instruments across time. Although the mean log likelihood of the test data is reasonable, the variance of these log likelihoods is too large, suggesting overfitting.

Instrument dimension and *2 dimension* supervised structures

The results obtained for the two last structures are displayed in fig 4.14. We compare a 2^{nd} order coupled Markov model with different forms of *instrument dimension* and *2 dimensions* supervised models. We use the two structures presented in fig 4.10(c) and 4.10(d). Then, using the changes of supervising structure presented in fig 4.11, we obtain four new models which are described in fig 4.14. The results show only modest improvement composed with on a low order coupled Markov model. This mainly comes from the poor trade-off between the complexity of these models and the amount of information carried by such added nodes. Let us take the example of the bass drum

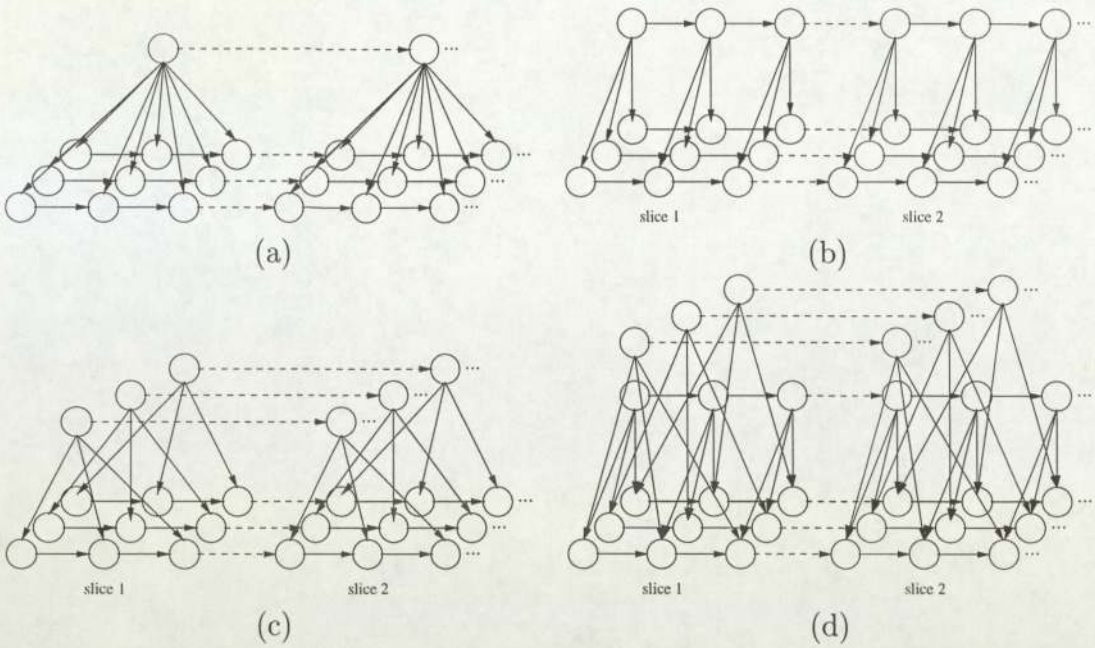


Figure 4.10: (a) *One node* supervised coupled Markov model. (b) *Time dimension* supervised coupled Markov model. (c) *Instrument dimension* supervised coupled Markov model. (d) *2 dimensions* supervised coupled Markov model. In order to simplify the notation, the inter connections of the CMM are not represented

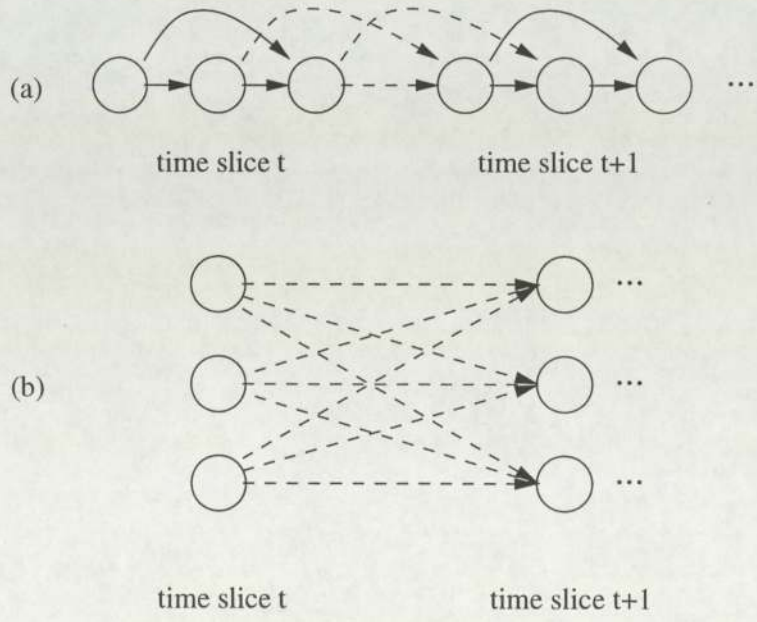


Figure 4.11: This figure presents an extension of the supervising structures presented in fig 4.10. For some simplification in the notation, we just drew the supervising structure. (a) second *time dimension* model, the top nodes are following a higher order Markov process, 2^{nd} in this figure. (b) second *instrument dimension* model where the supervising nodes are fully connected across time. The model presented in fig 4.10(d) will use one of these structural extensions, both or none

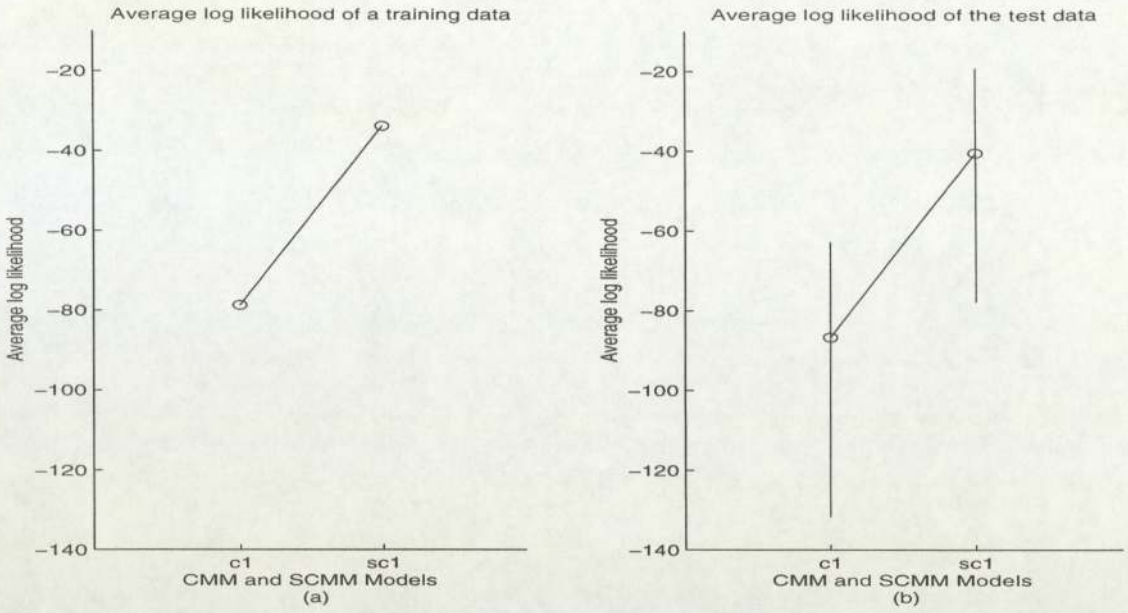


Figure 4.12: Comparison of a coupled Markov model (CMM) of slice width 1 and a *one node* supervised coupled Markov model (SCMM) of slice width 1. Because of the slice width, the order of the CMM is then 1. The number of hidden states modelled by the top node is set to 4.

in a 2^{nd} order CMM. The *Instrument dimension* supervising structure, fig 4.10(c), is redundant since a higher order Markov chain per instrument is enough to capture the local information of each couple $\{bd^{(t)}, bd^{(t+1)}\}$. The small gain in the approximation of the underlying process is largely counterbalanced by the complexity of the structure in a slice, leading to high computation time because of the large size of the cliques. The *2 dimension* supervised structure, partly based on the structure discussed above and displayed in fig 4.10(d), shows the same poor results.

In fig 4.14, we see that our supervised structures present a better fitting of the training set than the normal CMM. However, the two last ones provide an average log likelihood for a training data lower than $sv1$, the training test shows that the gain is not significant. Regarding the difference in time computation, ten times the time spent to train the 4^{th} order CMM, we should not consider using this kind of model.

Considering a Coupled Markov Model and its associated supervised model, it appears that not all supervising structures improve the capture of both local and global information. The *time dimension* supervised models show better results than the others. Due to its ability to model a stochastic process on the combination of instruments played at the same time and the possibility to increase the order of this process, order of the top nodes Markov chain in figures 4.10(b) and 4.11(a) for example, we will use it as our model in the next part. In fig 3.10, we saw the results of different hidden Markov models with respect to their number of hidden nodes. The comparison we made between the structure of HMMs and *time dimension* supervised models encourages us to consider the choice of the number of hidden nodes, finding a trade-off between the complexity of the model and its capability to capture global structure. We set the size of the top nodes to 6. A sample from a *one node* supervised model

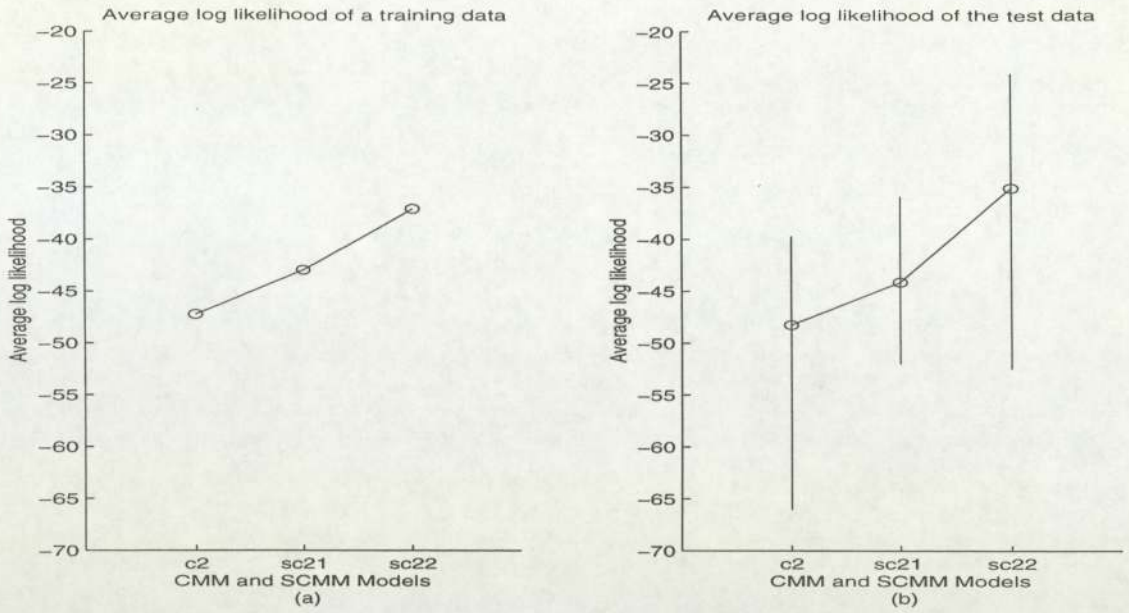


Figure 4.13: Comparison of a 2^{nd} order coupled Markov model and two *time dimension* supervised coupled Markov model, first and second order for the supervising structure. The first order one is presented in fig 4.10(b), by second order we mean 2^{nd} order on the Markov chain formed by the supervising nodes.

extended with the architecture shown in fig 4.11(a) with 6 hidden states is given in *sml_2_2_6_sample2.wav*. A part of the sequence is displayed in fig 4.15.

4.2 Marginalizing and Clamping

An advantage of attributing a stochastic process to each instrument is that we can work on its distribution inside the joint distribution of all the instruments represented by the whole model. We discuss here two main interesting uses of such a structure: firstly, marginalizing the coupled Markov model with respect to one instrument, we extract the information about how to play fewer instruments, following the same underlying, global process. Secondly, clamping the nodes of the subprocess of the instrument I_k, the model represents a new distribution of sequences over the rest of the instruments given what is played for I_k. In figure 4.16, we show a graphical representation of both tasks as applied to the coupled Markov models.

Marginalizing

Let us consider the instrument whose process is the first row of our coupled Markov model, the bass drum. We want to marginalize the drum sequences distribution represented by the model in fig 4.4 with respect to this instrument. This means summing over all states of the bass drum in order to produce a joint distribution on the remaining instruments. In this first order CMM, the CPT A associated with the second node of the second row defines how to play snare drum, sd, at time $t = 2$, depending on how

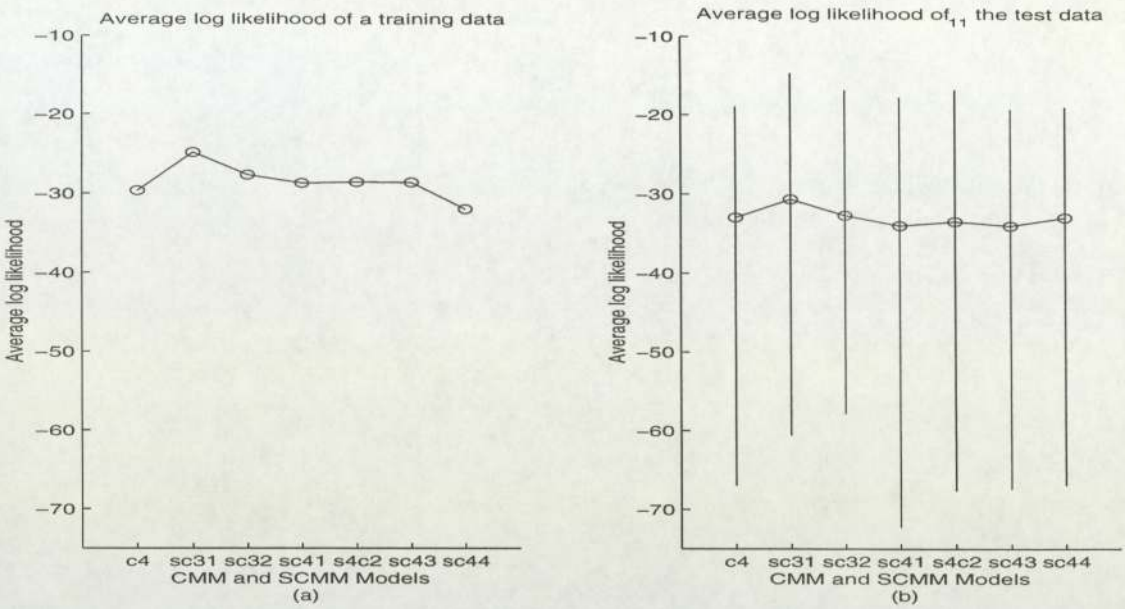


Figure 4.14: Comparison of a 4^{th} order coupled Markov model with 2 *instrument dimension* supervised coupled Markov models and a *2 dimension* one. The first supervised structure, labeled $s31$, is shown in fig 4.10(c). The one labeled $s32$ is basically the same except that all the supervising nodes interact with each other across time, fig 4.11(b). The third one, $s41$ is presented in fig 4.10(d). The three last models are a combination of $s41$, using the supervising structures presented in fig 4.11. $s42$, $s43$ and $s44$ are using the extensions in fig 4.11, respectively (a), (b) and (a)+(b)

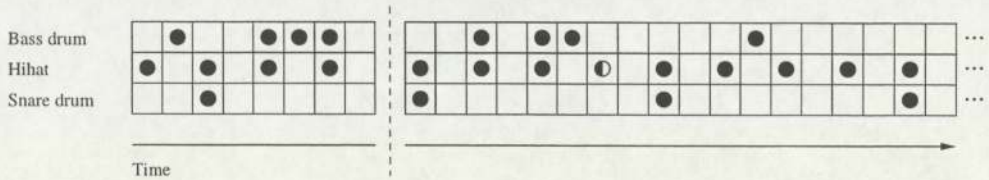


Figure 4.15: Part of a sequence generated from a supervised model, its type is shown in fig 4.10(b) and its structural extension in fig 4.11. We set the number of hidden nodes to 6 for the reasons explained above.

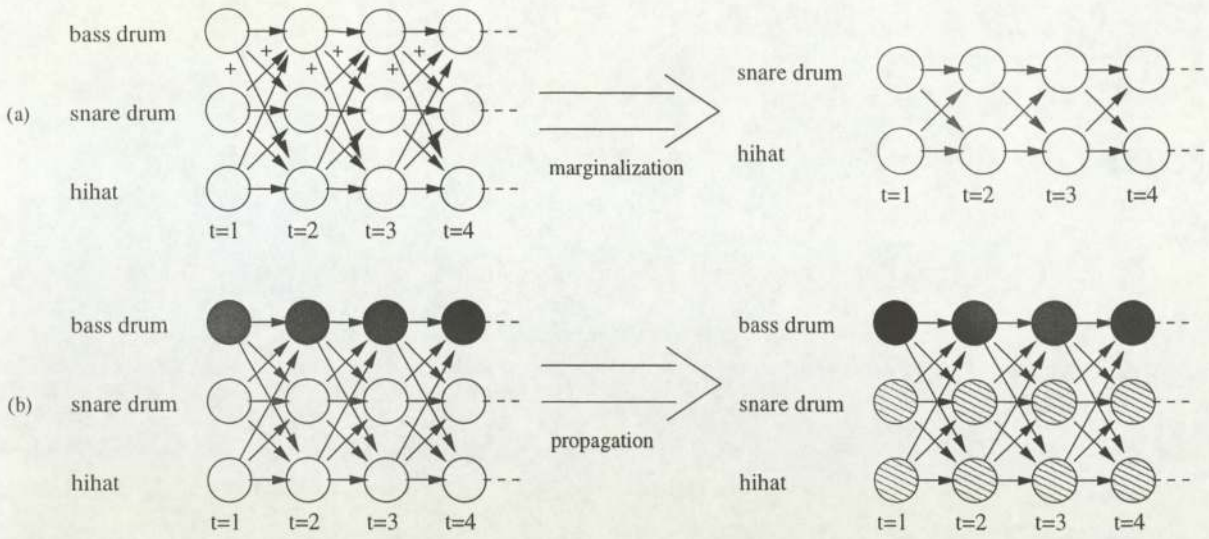


Figure 4.16: using coupled Markov models, we can perform two tasks: (a) *marginalizing* the model with respect to one instrument, therefore the coupled Markov model represents the underlying process of the reduced set of instruments, no matter what is played with the deleted one. (b) *clamping* an instrument nodes then propagate this extra evidences through the network to update the other CPTs

we played the three instruments at time $t = 1$

$$A = p(sd^{(2)}|bd^{(1)}, sd^{(1)}, hh^{(1)}) \tag{4.1}$$

Now, imagine we want to play snare drum and hihat (hh) without bass drum. This can be performed by summing this CPT over all the possible value of $bassdrum^{(1)}$

$$A^{\star} = \sum_{bd^{(1)}} p(sd^{(2)}|bd^{(1)}, sd^{(1)}, hh^{(1)}) \tag{4.2}$$

$$\propto p(sd^{(2)}|sd^{(1)}, hh^{(1)}) \tag{4.3}$$

Therefore, to obtain a model representing the distribution of $\{sd,hh\}$ sequences from our initial coupled Markov model, we sum over the bass drum values every CPT having a node from the bass drum process as a parent. When a node has a child in the first row of the model, we remove the link as if there was no process to influence.

Clamping

In clamping, we are interested in generating a drum sequence given the score for one or more of the instruments. As in section 2.2, clamping corresponds to dealing with evidence. The effect of clamping can be carried out automatically by a simple modification of the Junction Tree algorithm [10], using the Junction Tree form of its slice structure. For example

- Considering a full observation of the bass drum values, we clamp all the nodes associated with this instrument, the first row in the coupled Markov model. Each node has now a fixed value, see fig 4.16(b).

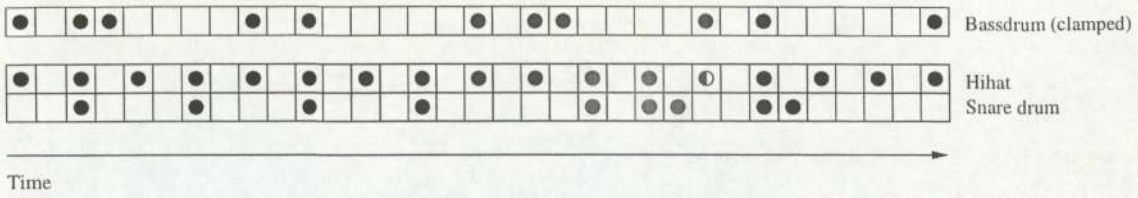


Figure 4.17: We show here the sequence sampled from a *time dimension* supervising structure. We set the order of the top node Markov process to 2 and the number of hidden nodes to 6. Then we clamped the nodes associated with the bass drum and recompute the CPTs of the other nodes of the model. The new model represents now the distribution of drum sequences given the bass drum sequence, sampling from it will complete the drum observations with a suitable sequence for the snare drum and the hihat.

- We propagate the new evidence using the Junction Tree algorithm.

The CMM that results from these two steps is now a graphical representation of the drum sequences distribution where the bass drum is played in a fixed way. Sampling from this distribution will produce sequences with the required bass drum rhythm and an appropriate completion for the two other instruments. An example is displayed in fig 4.17. We clamped the bass drum to the sequence presented in the upper line of the figure. The generated sequence of the {hihat, snare drum} instruments, conditional on the given bass drum sequence is plotted in the lower lines. The corresponding wav files are given in *clamp_sequence1.wav*, the bass drum sequence alone, and *clamp_sample1.wav*.

Chapter 5

Conclusion

Music is a very complex process. Through musical scores, the composers present their interpretation of music in a discrete way. This representation allows us to see a piece of music as a succession of discrete states. A deterministic approach is inappropriate for creating novel musical sequences, and we were therefore interested in a probabilistic approach. We make the assumption that the sequence of notes from a musical score is generated from a stochastic, dynamic underlying process.

This thesis attempted to determine in which way graphical models can be used in order to capture the structure of some pieces of music. We focused on the rhythm part as the simplest subprocess of music. We trained models to capture the structure of a sequence of instruments extracted from a drum textbook. Rhythm is a multi-level process, a note is part of a bar which is part of a song, all the notes have a role to play, even the silence. In order to model rhythm, we need to adopt the same structure and think of a multi-level model. The supervised structure introduced in the last chapter is, therefore, a first step to build a higher level process. The results obtained are encouraging but we were limited by two things: the data set and the computational complexity.

Since our data set is composed of basic drum sequences, we were limited in the complexity of our representation of rhythm. Moreover, we worked with sequences of basic drum exercises, meaning that there was no information about how to start a piece of rhythm or how to end it. However, the clamping procedure allows us to constrain the global information of the sequence. If we have training examples and sequences from the beginning, middle and end of pop songs, we could additionally label these sequences using an additional variable. Clamping this variable in certain states throughout the sequence, we should be able to constrain the global structure of a song adequately. However, the clamping procedure can lead to burdensome computational problems, and rapidly becomes infeasible. An approach to resolve this would be to consider approximate algorithms for computation with graphical models. Similarly, considering modelling more complex musical sequences with more instruments necessitates the use of more complex models. This should almost certainly require the use of approximation techniques.

Bibliography

[1] Xavier Boyen and Daphne Koller. Approximate learning of dynamic models. *Proceedings of the Eleventh Conference on Advances in Neural Information Processing Systems*, 1998.

[2] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, 1998.

[3] Matthew Brand. Coupled hidden Markov models for modeling interacting processes. Technical Report 405, MIT Media Lab Perceptual Computing, June 1997.

[4] Enrique Castillo, Jos Manuel Gutirrez, and Ali S. Hadi. *Expert Systems and Probabilistic Models.* Springer, 1997.

[5] Zoubin Gharamani. Learning dynamic Bayesian networks. *Adaptative Processing of temporal information*, october 1997.

[6] Zoubin Gharamani and Michael I. Jordan. Factorial hidden Markov models. *Machine Learning*, (29):245–275, 1997.

[7] Michael I. Jordan. *Learning in Graphical Models.* Kluwer, 1998.

[8] Michael I. Jordan, Zoubin Gharamani, and Lawrence K. Saul. Hidden Markov decision trees. Technical Report 9605, MIT Computational Cognitive Science.

[9] David Kulp, David Haussler, Martin G. Reese, and Frank H. Eeckman. A generalized HMM for the recognition of human genes in DNA. *ISMB*, 1996.

[10] Kevin P. Murphy. Filtering, smoothing and the Junction Tree Algorithm. 1998.

[11] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected application in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[12] Xavier Rodet. Musical sound signal analysis/synthesis: Sinusoidal+residual and elementary waveform models. *IEEE Time-Frequency and Time-Scale Workshop*, 1997.

[13] Uffe Kjærulff. A computational scheme for dynamic Bayesian networks. Technical report, Department of Mathematics and Computer Science Aalborg University, April 1993.

[14] Christopher K. I. Williams and Xiaojuan Feng. Tree-structured belief networks as models of images. *ICANN*, 1999.