Chapter 8

Algorithms

Giovanni De Poli

8.1 Markov Models and Hidden Markov Models

Andrei Andreevich Markov first introduced his mathematical model of dependence, now known as Markov chains, in 1907. A Markov model (or Markov chain) is a mathematical model used to represent the tendency of one event to follow another event, or even to follow an entire sequence of events. Markov chains are matrices comprised of probabilities that reflect the dependence of one or more events on previous events. Markov first applied his modeling technique to determine tendencies found in Russian spelling. Since then, Markov chains have been used as a modeling technique for a wide variety of applications ranging from weather systems to baseball games.

Statistical methods of Markov source or hidden Markov modeling (HMM) have become increasingly popular in the last several years. The models are very rich in mathematical structure and hence can form a basis for use in a wide range of applications. Moreover the models, when applied properly, work very well in practice for several important applications.

8.1.1 Markov Models or Markov chains

Markov models are very useful to represent families of sequences with certain specific statistical properties. To explain the idea consider a simple 3 state model of the weather. We assume that once a day, the weather is observed as being one of the following: rain (state 1); cloudy (state 2); sunny (state 3).

If we examine a sequence of observation during a month, the state rain appears a few times, and it can be followed by rain, cloud or sun. Given a long sequence of observations, we can count the number of times the state rain is followed by, say, a cloudy state. From this we can estimate the probability that a rain is followed by a cloudy state. If this probability is 0.3 for example, we indicate it as shown in Figure 8.1. The figure also shows examples of probabilities for every state to transition

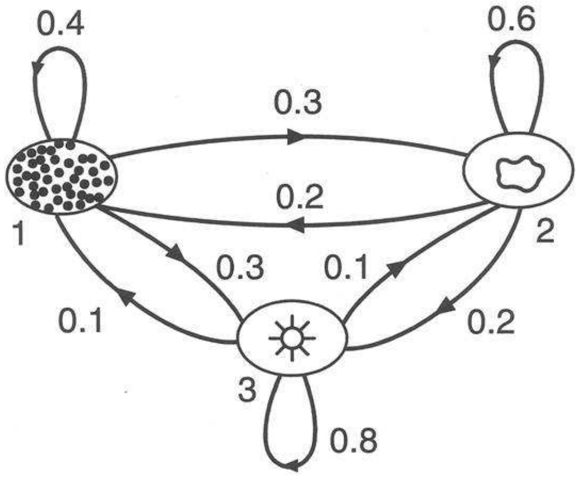


Figure 8.1: State transition of the weather Markov model (from Rabiner 1999).

to other states, including itself. The first row of the matrix A

$$A = \{a_{i,j}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \tag{8.1}$$

shows the three probabilities more compactly (notice that their sum is unity). Similarly the probabilities that the cloudy state would transition into the three states can be estimated, and is shown in the second row of the matrix. This 3×3 matrix is called a state transition matrix, and is denoted as $\mathbf{A}$ and the coefficients have the properties $a_{i,j} \geq 0$ and $\sum_j a_{i,j} = 1$ since they obey standard stochastic constraints. Figure 8.1 is called a Markov model.

Formally a Markov model (MM) models a process that goes through a sequence of discrete states, such as notes in a melody. At regular spaced, discrete times, the system undergoes a change of state (possibly back to same state) according to a set of probabilities associated with the state. The time instances for a state change is denoted t and the actual state at time t as $x(t)$. The model is a weighted automaton that consists of:

- A set of N states, $S = \{s_1, s_2, s_3, \ldots, s_N\}$.
- A set of transition probabilities, $\mathbf{A}$, where each $a_{i,j}$ in $\mathbf{A}$ represents the probability of a transition from s_i to s_j. I.e $a_{i,j} = P\left[x(t) = j \,|\, x(t-1) = i\right]$.
- A probability distribution, π, where π_i is the probability the automaton will begin in state s_i, i.e $\pi_i = P(x_1 = i)$, $1 \leq i \leq N$. Notice that the stochastic property for the initial state distribution vector is $\sum_i \pi_i = 1$.
- E, a subset of S containing the legal ending states.

In this model, the probability of transitioning from a given state to another state is assumed to depend only on the current state. This is known as the Markov property.

Given a sequence or a set of sequences of similar kind (e.g., a long list of melodies from a composer) the parameters of the model (the transition probabilities) can readily be estimated. The process of identifying the model parameters is called training the model. In all discussions it is implicitly assumed that the probabilities of transitions are fixed and do not depend on past transitions.

Suppose we are given a Markov model (i.e., $\mathbf{A}$ given). Given an arbitrary state sequence $\mathbf{x} = [x(1), x(2), ..., x(L)]$ we can calculate the probability that **x** has been generated by our model. This is given by the product

$$P(\mathbf{x}) = P(x(1)) \times P(x(1) \to x(2)) \times P(x(2) \to x(3)) \times \cdots \times P(x(L-1) \to x(L))$$

where $P(x(1)) = \pi(x(1))$ is the probability that x(1) is the initial state, $P(x(k) \to x(m))$ is the transition probability for going from $x(k)$ to $x(m)$, and can be found from the matrix $\mathbf{A}$. For example with reference to the weather Markov model of equation 8.1, given that the weather on day 1 is sunny (state 3), we can ask the question: What is the probability that the weather for next 7 days will be "sun-sun-rai-rain-sun-cloudy-sun . ."? This probability can be evaluated as

$$\begin{aligned} P &= \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{13} \\ &= 1(0.8)(0.8)(0.1)(0.4)(0.3)(0.1)(0.2) \\ &= 1.536 \times 10^{-4} \end{aligned}$$

The usefulness of such computation is as follows: given a number of Markov models ($\mathbf{A_1}$ for a composer, $\mathbf{A_2}$ for a second composer, and so forth) and given a melody $\mathbf{x}$, we can calculate the probabilities that this melody is generated by any of these models. The model which gives the highest probability is most likely the model which generated the sequence.

8.1.2 Hidden Markov Models

A hidden Markov model (HMM) is obtained by a slight modification of the Markov model. Thus consider the state diagram shown in Figure 8.1 which shows three states numbered 1, 2, and 3. The probabilities of transitions from the states are also indicated, resulting in the state transition matrix $\mathbf{A}$ shown in equation 8.1. Now we can suppose that we can not observe directly the state, but only a symbol that is associated in a probabilistic way to the state. For example when the weather system is in a particular state, it can output one of four possible symbols L, M, H, VH (corresponding to temperature classes low, medium, high, very high), and there is a probability associated with each of these. This is summarized in the so-called output matrix $\mathbf{B}$

$$\mathbf{B} = \{b_{i,j}\} = \begin{bmatrix} 0.4 & 0.3 & 0.2 & 0.1 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.4 & 0.4 \end{bmatrix} \tag{8.2}$$

The element $b_{i,j}$ represents the probability of observing the temperature class j when the weather is in the (non observable) state i, i.e. $b_{i,j} = P(x(t) = s_i | x(t) = j)$. For example when the weather is rainy (state $i = 1$), the probability of measuring medium temperature (output symbol $j = 2$) is $b_{1,2} = 0.3$.

More formally, an HMM requires two things in addition to that required for a standard Markov model:

- A set of possible observations, $O = \{o_1, o_2, o_3, \ldots, o_n\}$.
- A probability distribution $\mathbf{B}$ over the set of observations for each state in S.

Basic HMM problems In order to apply the hidden Markov model theory successfully there are three problems that need to be solved in practice. These are listed below along with names of standard algorithms which have been developed for these.

1. Learn structure problem. Given an HMM (i.e., given the matrices $\mathbf{A}$ and $\mathbf{B}$) and an output sequence $o(1), o(2), \ldots$, compute the state sequence $x(k)$ which most likely generated it. This is solved by the famous Viterbi's algorithm (see 8.1.4.2).

2. Evaluation or scoring problem. Given the HMM and an output sequence $o(1), o(2), \ldots$ compute the probability that the HMM generates this. We can also view the problem as one of scoring how well a given model matches a given output sequence. If we are trying to choose among several competing models, this ranking allow us to choose the model that best matches the observations. The forward-backward algorithm solves this (see 8.1.4.1).

3. Training problem. How should one design the model parameters $\mathbf{A}$ and $\mathbf{B}$ such that they are optimal for an application, e.g., to represent a melody? The most popular algorithm for this is the expectation maximization algorithm commonly known as the EM algorithm or the Baum-Welch algorithm (see Rabiner [1989] for more details).

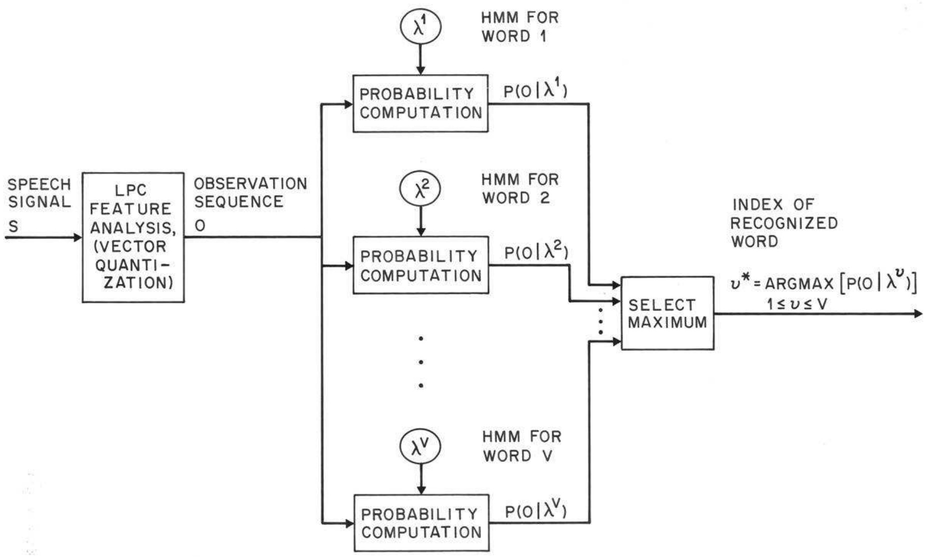


Figure 8.2: Block diagram of an isolated word recognizer (from Rabiner 1999).

For example let us consider a simple isolated word recognizer (see Figure 8.2). For each word we want to design a separate N-state HHM. We represent the speech signal as a time sequence of coded spectral vectors. Hence each observation is the index of the spectral vector closest to the original speech signal. Thus for each word, we have a training sequence consisting of repetitions of codebook indices of the word.

The first task is to build individual word models. This task is done by using the solution to Problem 3 to estimate model parameters for each word model. To develop an understanding of physical meaning of the model state, we use the solution to Problem 1 to segment each of the word state sequence into states, and then study the properties of the spectral vectors that lead to the observations occurring in each state. Finally, once the set of HMMs has been designed, recognition of an unknown word is performed using the solution to Problem 2 to score each word model based on the observation sequence, and select the word whose model score is highest.

We should remark that the HMM is a stochastic approach which models the given problem as a doubly stochastic process in which the observed data are thought to be the result of having passed the true (hidden) process through a second process. Both processes are to be characterized using only the one that could be observed. The problem with this approach, is that one do not know anything about the Markov chains that generate the speech. The number of states in the model is unknown, there probabilistic functions are unknown and one can not tell from which state an observation was produced. These properties are hidden, and thereby the name hidden Markov model.

8.1.3 Markov Models Applied to Music

Hiller and Isaacson (1957) were the first to implement Markov chains in a musical application. They developed a computer program that used Markov chains to compose a string quartet comprised of four movements entitled the Illiac Suite. Around the same time period, Meyer and Xenakis (1971) realized that Markov chains could reasonably represent musical events. In his book Formalized Music Xenakis [1971], Xenakis described musical events in terms of three components: frequency, duration, and intensity. These three components were combined in the form of a vector and then were used as the states in Markov chains. In congruence with Xenakis, Jones (1981) suggested the use of vectors to describe notes (e.g., note = pitch, duration, amplitude, instrument) for the purposes of eliciting more complex musical behavior from a Markov chain. In addition, Polansky, Rosenboom, and Burk (1987) proposed the use of hierarchical Markov chains to generate different levels of musical organization (e.g., a high level chain to define the key or tempo, an intermediate level chain to select a phrase of notes, and a low level chain to determine the specific pitches). All of the aforementioned research deals with the compositional aspects and uses of Markov chains. That is, all of this research was focused on creating musical output using Markov chains.

8.1.3.1 HMM models for music search: MuseArt

In the MuseArt system for music search and retrieval, developed at Michigan University by Jonah Shifrin, Bryan Pardo, Colin Meek, William Birmingham, musical themes are represented using a hidden Markov model (HMM).

Representation of a query. The query is treated as an observation sequence and a theme is judged similar to the query if the associated HMM has a high likelihood of generating the query. A piece of music is deemed a good match if at least one theme from that piece is similar to the query. The pieces are returned to the user in order, ranked by similarity.

A query is a melodic fragment sung by a single individual. The singer is asked to select one syllable, such as ta or la, and use it consistently during the query. The consistent use of a single consonant-vowel pairing lessens pitch-tracker error by providing a clear onset point for each note, as well as reducing error caused by vocalic variation. A query is recorded as a .wav file and is transcribed into a MIDI based representation using a pitch-tracking system. Figure 8.3 shows a time-amplitude representation of a sung query, along with example pitch-tracker output (shown as piano roll) and a sequence of values derived from the MIDI representation (the $deltaPitch$, IOI and $IOIratio$ values). Time values in the figure are rounded to the nearest 100 milliseconds. We define the following.

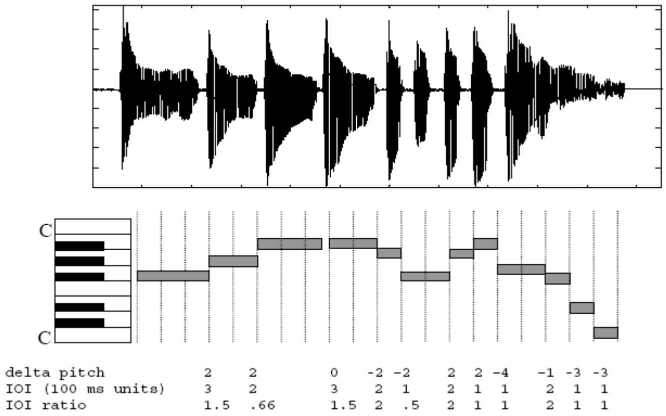


Figure 8.3: A sung query (from Shifrin 2002)

- A note transition between note n and note $n+1$ is described by the duple $< deltaPitch$, $IOIratio >$.
- $deltaPitch_n$ is the musical interval, i.e. the pitch difference in semitones between note n and note $n+1$.
- $IOIratio_n$ is IOI_n/IOI_{n+1}, where the inter onset interval (IOI_n) is the difference between the onset of notes n and $n+1$. For the final transition, $IOI_n = IOI_n/duration_{n+1}$.

A query is represented as a sequence of note transitions. Note transitions are useful because they are robust in the face of transposition and tempo changes. The $deltaPitch$ component of a note transition captures pitch-change information. Two versions of a piece played in two different keys have the same $deltaPitch$ values. The $IOIratio$ represents the rhythmic component of a piece. This remains constant even when two performances are played at very different speeds, as long as relative durations within each performance remain the same. In order to reduce The number of possible IOI ratios is reduced by quantizing them to one of 27 values, spaced evenly on a logarithmic scale. A logarithmic scale was selected because data from a pilot study indicated that sung $IOIratio$ values fall naturally into evenly spaced bins in the log domain.

The directed graph in Figure 8.4 represents a Markov model of a scalar passage of music. States are note transitions. Nodes represent states. The numerical value below each state indicates the probability a traversal of the graph will begin in that state. As a default, all states are assumed to be legal ending states. Directed edges represent transitions. Numerical values by edges indicate transition probabilities. Only transitions with non-zero probabilities are shown.

In Markov model, it is implicitly assumed that whenever state s is reached, it is directly observable, with no chance for error. This is often not a realistic assumption. There are multiple possible sources of error in generating a query. The singer may have incorrect recall of the melody he or she is attempting to sing. There may be production errors (e.g., cracked notes, poor pitch control). The transcription system may introduce pitch errors, such as octave displacement, or timing errors due to the quantization of time. Such errors can be handled gracefully if a probability distribution over the set of possible observations (such as note transitions in a query) given a state (the intended note transition of the singer) is maintained. Thus, to take into account these various types of errors, the Markov model should be extended to a hidden Markov Model, or HMM. The HMM allows us a probabilistic

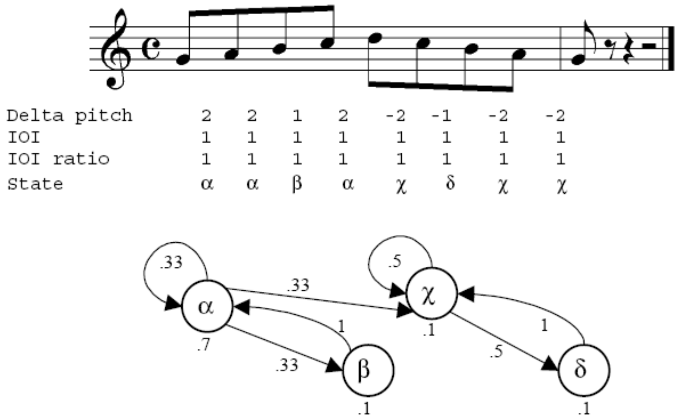


Figure 8.4: Markov model for a scalar passage (from Shifrin 2002)

map of observed states to states internal to the model (hidden states). In the system, a query is a sequence of observations. Each observation is a note-transition duple, $< deltaPitch, IOIratio >$. Musical themes are represented as hidden Markov models whose states also corresponds to note-transition duples. To make use of the strengths of a hidden Markov model, it is important to model the probability of each observation o_i in the set of possible observations, O, given a hidden state, s.

Making Markov Models from MIDI. Our system represents musical themes in a database as HMMs. Each HMM is built automatically from a MIDI file encoding the theme. The unique duples characterizing the note transitions found in the MIDI file form the states in the model. FigureFigure 8.4 shows a passage with eight note transitions characterized by four unique duples. Each unique duple is represented as a state. Once the states are determined for the model, transition probabilities between states are computed by calculating what proportion of the time state a follows state b in the theme. Often, this results in a large number of deterministic transitions. Figure 8.5 is an example of this, where only a single state has two possible transitions, one back to itself and the other on to the next state. Note that there is not a one-to-one correspondence between model and observation se-

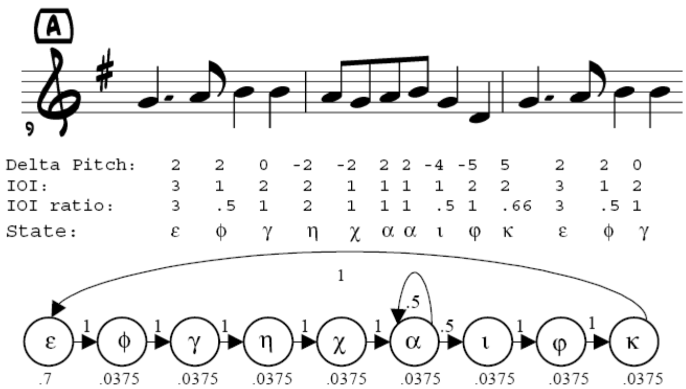


Figure 8.5: Markov model for Alouette fragment (from Shifrin 2002)

quence. A single model may create a variety of observation sequences, and an observation sequence may be generated by more than one model. Recall that our approach defines an observation as a duple, ¡deltaPitch, IOIratio¿. Given this, the observation sequence $q = \{(2,1),(2,1),(2,1)\}$ may be generated by the HMM in Figure 8.4 or the HMM in Figure 8.5.

Finding the best target. The themes in the database are coded as HMMs and the query is treated as an observation sequence. Given this, we are interested in finding the HMM most likely to generate the observation sequence. This can be done using the Forward algorithm. The Forward algorithm, given an HMM and an observation sequence, returns a value between 0 and 1, indicating the probability the HMM generated the observation sequence. Given a maximum path length, L, the algorithm takes all paths through the model of up to L steps. The probability each path has of generating the observation sequence is calculated and the sum of these probabilities gives the probability that the model generated the observation sequence. This algorithm takes on the order of $|S|^2L$ steps to compute the probability, where $|S|$ is the number of states in the model.

Let there be an observation sequence (query), O, and a set of models (themes), M. An order may be imposed on M by performing the Forward algorithm on each model m in M and then ordering the set by the value returned, placing higher values before lower. The i-th model in the ordered set is then the i-th most likely to have generated the observation sequence. We take this rank order to be a direct measure of the relative similarity between a theme and a query. Thus, the first theme is the one most similar to the query.

8.1.3.2 Markov sequence generator

Markov models can be thought of as generative models. A generative model describes an underlying structure able to generate the sequence of observed events, called an observation sequence. Note that there is not a one-to-one correspondence between model and observation sequence. A single model may create a variety of observation sequences, and an observation sequence may be generated by more than one model.

A HMM can be used as generator to give an observation sequence O as follow

1. Choose initial state $x(1) = S_1$ according the initial state distribution π.
2. Set $t = 1$
3. Choose $o(t)$ according the symbol probability distribution in state $x(t)$ described in matrix $\mathbf{B}$
4. Transit to new state $x(t+1) = S_j$ according to the state transition probability for state $x(t) = i$, i.e. $a_{i,j}$
5. Set $t = t + 1$ and return to step 2

If a simple Markov model is used as generator, step 3 is skipped, and the state $x(t)$ is used in output.

The ”hymn tunes” of Figure 8.6 were generated by computer from an analysis of the probabilities of notes occurring in various hymns. A set of hymn melodies were encoded (all in C major). Only hymn melodies in 4/4 meter and containing two four-bar phrases were used. The first ”tune” was generated by simply randomly selecting notes from each of the corresponding points in the analyzed melodies. Since the most common note at the end of each phrase was ‘C’ there is a strong likelihood that the randomly selected pitch ending each phrase is C.

Figure 8.6: "Hymn tunes" generated by computer from an analysis of the probabilities of notes occurring in various hymns. From Brooks, Hopkins, Neumann, Wright. "An experiment in musical composition." IRE Transactions on Electronic Computers, Vol. 6, No. 1 (1957).

8.1.4 Algorithms

8.1.4.1 Forward algorithm

The Forward algorithm is uses to solve the evaluation or scoring problem. Given the HMM $\lambda = (A, B, \Pi)$ and an observation sequence $O = o(1)o(2)\ldots o(L)$ compute the probability $P(O|\lambda)$ that the HMM generates this. We can also view the problem as one of scoring how well a given model matches a given output sequence. If we are trying to choose among several competing models, this ranking allow us to choose the model that best matches the observations. The most straighforward procedure is through enumerating every possible state sequence of lenght L (the number of observations), computing the joint probability of the state sequence and O and finally summing the joint probabilty over all possible sate sequence. But if there are N possible states that can be reached, there are N^L possible state sequences and thus such direct approach have exponential computational complexity.

However we can notice that there are only N states and we can apply a dynamic programming stategy. To this purpose let us define the forward variable $\alpha_t(i)$ as

$$\alpha_t(i) = P(o(1)o(2)\ldots o(t),\, x(t) = s_i|\lambda)$$

i.e. the probability of the partial observation $o(1)o(2)\ldots o(t)$ and state s_i at time t, given the model λ. The Forward algorithm solves the problem with a dynamic programming strategy, using an iteration on the sequence length (time t), as follows:

1. Initialization

$$\alpha_1(i) = \pi(i)b_i(o_1),\ 1 \le i \le N$$

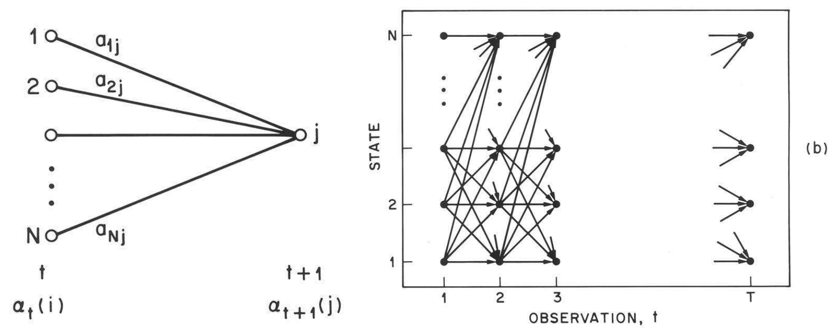
2. Induction

$$\alpha_{t+1}(i) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}) \quad 1 \le t \le L-1$$
$$1 \le i \le N$$

3. Termination

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_L(i)$$

Step 1) initializes the forward probabilities as the joint probability of state i and initial observation $o(1)$. The induction step is illustrated in Figure 8.7(a). This figure shoes that state s_j can be reached at time $t+1$ from the N possible states at time t. Since $\alpha_t(i)$ is the probability that $o(1)o(2)\dots o(t)$ is observed and $x(t) = s_i$, the product $\alpha_t(i)a_{ij}$ is the probability that $o(1)o(2)\dots o(t)$ is observed and state s_j is reached at time $t+1$ via state s_i at time t. Summing this product over all the possible states results in the probability of s_j with all the previous observations. Finally $\alpha_{t+1}(i)$ s obtained by accounting for observation o_{t+1} in state s_j, i.e. by multiplying by the probability $b_j(o_{t+1})$. Finally step 3) gives the desired $P(O|\lambda)$ as the sum of the terminal forward variables $\alpha_L(i)$. In fact $\alpha_L(i)$ is the probability of the observed sequence and that the system at time $t = L$ is in the state s_i. Hence $P(O|\lambda)$ is just the sum of the $\alpha_L(i)$'s. The time computational complexity of this algorithm is $O(N^2L)$. The forward probability calculation is based upon the lattice structure shown in figure 8.7(b). The key is that since there are only N states, all the possible state sequences will remerge into these N nodes, no matter how long the observation sequence. Notice that the calculation of $\alpha_t(i)$ involves multiplication with probabilities. All these probabilities have a value less than 1 (generally significantly less than 1), and as t starts to grow large, each term of $\alpha_t(i)$ starts to head exponentially to zero, exceed the precision range of the machine. To avoid this problem, a version of the Forward algorithm with scaling should be used. See Rabiner [1989] for more details.



(a) (b)

Figure 8.7: (a) Illustration of the sequence of operations required for the computation of the forward variable $\alpha_{t+1}(i)$. (b) Implementation of the computation of $\alpha_{t+1}(i)$ in terms of a lattice of observation t and states i.

8.1.4.2 Viterbi algorithm

The Viterbi algorithm, based on dynamic programming, is used to solve the structure learning problem. Given an HMM λ (i.e., given the matrices $\mathbf{A}$ and $\mathbf{B}$) and an output sequence $O = \{o(1)o(2)\dots o(L)\}$, find the $single$ best state sequence $X = \{x(1)x(2)\dots x(L\}$ which most likely generated it. To this purpose we define the quantity

$$\delta_t(i) = P\left[x(1)x(2)\dots x(t) = s_i, o(1)o(2)\dots o(t)\,|\lambda\right]$$

i.e. $\delta_t(i)$ is the best score (highest probability) along a single path at time t, which accounts for the first t observations and ends in state s_i. By induction we have

$$\delta_{t+1}(i) = \max_i \left[\delta_t(i)a_{ij}\right] b_j(o_{t+1})$$

To actually retrieve the state sequence, we need to keep track of the argument which maximized the previous expression, for each t and j using a predecessor array $\psi_t(j)$. The complete procedure of Viterbi algorithm is

1. Initialization
   for $1 \le i \le N$
   $\delta_1(i) = \pi(i)b_i(o_1)$
   $\psi_1(i) = 0$

2. Induction
   for $1 \le t \le L-1$
   for $1 \le j \le N$
   $\delta_{t+1}(j) = \max_i \left[\delta_t(i)a_{ij}\right] b_j(o_{t+1})$
   $\psi_{t+1}(j) = \text{argmax}_i \left[\delta_t(i)a_{ij}\right]$

3. Termination
   $P^* = \max_i \left[\delta_T(i)\right]$
   $x^*(T) = \text{argmax}_i \left[\delta_T(i)\right]$

4. Path backtracking
   for $t = L-1$ downto 1
   $x^*(t) = \psi_{t+1}(x^*_{t+1})$

Notice that the structure of Viterbi algorithm is similar in implementation to forward algorithm. The major difference is the maximization over the previous states which is used in place of the summing procedure in forward algorithm. Both algorithms used the lattice computational structure of figure 8.7(b) and have computational complexity N^2L. Also Viterbi algorithm presents the problem of multiplicationof probabilities. One way to avoid this is to take the logarithm of the model parameters, giving that the multiplications become additions. The induction thus becomes

$$\log[\delta_{t+1}(i)] = \max_i(\log\left[\delta_t(i)\right] + \log\left[a_{ij}\right] + \log\left[b_j(o_{t+1})\right]$$

Obviously will this logarithm become a problem when some model parameters has zeros present. This is often the case for A and π and can be avoided by adding a small number to the matrixes. See Rabiner [1989] for more details.

To get a better insight of how the Viterbi (and the alternative Viterbi) works, consider a model with $N = 3$ states and an observation of length $L = 8$. In the initialization ($t = 1$) is $\delta_1(1)$, $\delta_1(2)$ and $\delta_1(3)$ found. Lets assume that $\delta_1(2)$ is the maximum. Next time ($t = 2$) three variables will be used namely $\delta_2(1)$, $\delta_2(2)$ and $\delta_2(3)$. Lets assume that $\delta_2(1)$ is now the maximum. In the same manner will the following variables $\delta_3(3)$, $\delta_4(2)$, $\delta_5(2)$, $\delta_6(1)$, $\delta_7(3)$ and $\delta_8(3)$ be the maximum at their time, see Fig.8.8. This algorithm is an example of what is called the Breadth First Search (Viterbi employs this essentially). In fact it follows the principle: "Do not go to the next time instant $t+1$ until the nodes at at time T are all expanded".

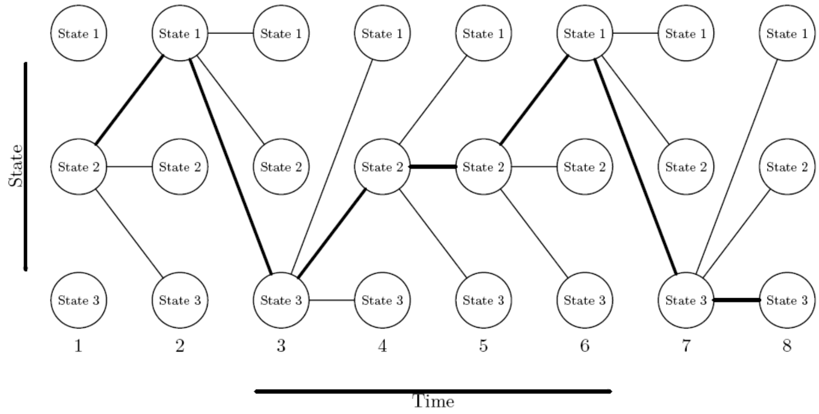


Figure 8.8: Example of Viterbi search.

8.2 Algorithms for music composition

Composers have long been fascinated by mathematical concepts in relation to music. The concept of ”music of the spheres,” dating back to Pythagoras, held the notion that humans were governed by the perfect proportions of the natural universe. This mathematical order may be seen in the musical interval choice and system of organization that was used by the ancient Greek culture.

Procedures that entail rules or provisions to govern the act of musical composition have been used since the Medieval period; these same principles have been applied in very specific methods to many of the recent computer programs developed for algorithmic composition.

8.2.1 Algorithmic Composition

Algorithmic composition could be described as ”a sequence (set) of rules (instructions, operations) for solving (accomplishing) a [particular] problem (task) [in a finite number of steps] of combining musical parts (things, elements) into a whole (composition)”. From this definition we can see that it is not necessary to use computers for algorithmic composition as we often infer; Mozart did not when he described the Musical Dice Game.

The concept of algorithmic composition is not something new. Pythagoras (around 500 B.C.) believed that music and mathematics were not separate studies. Hiller and Isaacson (1959) were probably the first who used a computational model using random number generators and Markov chains for algorithmic composition. Since then many researchers have tried to address the problem of algorithmic composition from different points of view.

Some of the algorithmic programs and compositions specify score information only. Score information includes pitch, duration, and dynamic material, whether written for acoustic and/or electronic instruments. That is, there are instances in which a composer makes use of a computer program to generate the score while the instrumental selection has been predetermined as either an electronic orchestra or a realization for acoustic instruments. Other algorithmic programs specify both score and electronic sound synthesis. In this instance, the program is used not only to generate the score, but also the electronic timbres to be used in performance. This distinction has its roots in the traditional differentiation between score and instrument, but a computer-generated continuum between two different sounds, however, is both score and sound synthesis. In both types of synthesis, the appearance of events in time is structured, both globally (form) as well as locally (sound, timbre).

The selection or construction of algorithms for musical applications can be divided into three categories:

- **Modeling traditional, non-algorithmic compositional procedures.** This category refers to algorithms that model traditional omposition techniques (tonal harmony, tonal rhythm, counterpoint rules, specific formal devices, serial parametrisation, etc.). This approach has been scarcely used in music composition, but it has become an essential element of musicological Research.
- **Modeling new, original compositional procedures, different from those known before.** This category refers to algorithms that create new constructs which sport some inherently musical qualities. These algorithms range from Markov chains to stochastic and probabilistic functions. Such algorithms have been pioneered by the composer Iannis Xenakis in the ”50s-”60s and widely used by a consistent group of composers since then.
- **Selecting algorithms from extra-musical disciplines.** This category refers to algorithms invented to model other, non-musical, processes and forms. Some of these algorithms have been used very proficiently by composers to create specific works. These algorithms are generally related to self-similarity (which is a characteristic that is closely related to that of ”thematic development” which seems to belong universally to all musics) and they range from genetic algorithms to fractal systems, from cellular automata, to swarming models and coevolution. In this same category, a persistent trend of using biological data to generate compositional structures has developed since the 60’s. Using brain activity (through EEG measurements), hormonal activity, human body dynamics, there has been a constant attempt to render biological data with musical structures.

8.2.2 Computer Assisted Composition

Another use of computers for music generation has been that of *Computer-Assisted Composition*. In this case, computers do not generate complete scores. Rather, they provide mediation tools to help composers to manage and control some aspects of musical creation. Such aspects may range from extremely relevant decision-making processes to minuscule details according to the composers’ wishes.

Two main approaches can be observed in Computer-Assisted Composition:

- Integrated tools and languages that will cover all possible composing desiderata;
- Personalised micro-programs written in small languages like awk, lisp, perl, prolog, python, ruby, etc. (written by the composer herself and possibly interconnected together via pipes, sockets and common databases).

While computer assistance may be a more practical and less generative use of computers in Musical Composition, it is currently enjoying a much wider diffusion among composers.

8.2.3 Categories of algorithmic processes

A review can not be exhaustive because there have been so many attempts. In the following subsections [1] we give some representative examples of systems which employ different methods which we categorise, based on their most prominent feature, as follows:

[1] adapted from Papadopoulos, Wiggins 1993

8.2.3.1 Mathematical models

Stochastic processes and especially Markov chains have been used extensively in the past for algorithmic composition (e.g., Xenakis, 1971).

The basic algorithm is

Algorithm `GenerateAndTest`

while composition is not terminated
  generate raw materials
  modify according to various functions
  select the best results according to rules

The simplest way to generate music from a history-based model is to sample, at each stage, a random event from the distribution of events at that stage. After an event is sampled, it is added to the piece, and the process continues until a specified piece duration is exceeded.

Algorithm `RandomWalk`

Get events distribution by analysing a music repertoire
while composition is not terminated
  sample a random event from the distribution of events
  add to the piece

One manner of statistical analysis that has been frequently used in musical composition is Markov Analysis or Markov Chains. Named for the mathematician Andre Andreevich Markov (1856-1922), Markov Chains were conceived as a means by which decisions could be made based on probability. Information is linked together in a series of states based on the probability that state A will be followed by state B. The process is continually in transition because state A is then replaced by state B which continues to look at the probability of being followed by yet another state B. The so-called orders of Markov Analysis indicate the relationship between states. For instance, zeroth-order analysis assumes that there are no relationships between states; that is, the relationship between any two states is random. First-order analysis simply counts the frequency with which specific states occur within the given data. Second-order analysis examines the relationships between any two consecutive states (e.g., what is the probability that the state B would follow state A?). Third-order analysis determines the probability of three consecutive states occurring in a row (e.g., what is the probability that state A would be followed by state B, would be followed by state C?). Fourth-order analyzes the chance of four states following each other. Composer/scientist Lejaren Hiller made use of Markov Chains, statistical analysis, and stochastic procedures in algorithmic composition beginning in the late 1950s.

Probably the most important reason for stochastic precesses is their low complexity which makes them good candidates for real-time applications.

We also see computational models based on chaotic nonlinear systems or iterated functions but it is difficult to judge the quality of their output, because, unlike all the other approaches, their "knowledge" about music is not derived from humans or human works. Since the 1970s basic principles of the irregularities in nature have been studied by the scientific community, and by the 1980s chaos was the focus of a great deal of attention. The new science has spawned its own language, an elegant shop talk of fractals and bifurcations, intermittencies and periodicities, folded-towel diffeomorphisms and smooth noodle maps. . . To some physicists chaos is a science of process rather than state, or becoming rather than being. One subcategory of chaotic structures that has come to the forefront since ca. 1975 is fractals. Fractals are recursive and produce 'parent-child' relationships in which the offspring replicate the initial structure. Seen in visual art as smaller and smaller offshoots from the original stem, fractals were categorized by Benoit Mandelbrot in his book, The Fractal Geometry of Nature. The underlying principles of chaos may best be thought of in terms of natural, seemingly disorderly designs.

> "Nature forms patterns. Some are orderly in space but disorderly in time, others orderly in time but disorderly in space. Some patterns are fractal, exhibiting structures self-similar in scale. Others give rise to steady states or oscillating ones. Pattern formation has become a branch of physics and of materials science, allowing scientists to model the aggregation of particles into clusters, the fractured spread of electrical discharges, and the growth of crystals in ice and metal alloys."

The main disadvantages of stochastic processes are:

- Someone needs to find the probabilities by analysing many pieces. Something necessary if we want to simulate one style. The resulting models will only generate music of similar style to the input.
- For higher order Markov models, transition tables become unmanageably large for the average computer. While many techniques exist for a more compact representation of sparse matrices (which usually result for higher order models), these require extra computational effort that can hinder real time performance.
- The deviations from the norm and how they are incorporated in the music is an important aspect. They also provide little support for structure at higher levels (unless multiple layered models are used where each event represents an entire Markov model in itself).

8.2.3.2 Knowledge based systems

Many early systems focused on taking existing musicological rules and embedding them in computational procedures. In one sense, most AI systems are knowledge based systems (KBS). Here, we mean systems which are symbolic and use rules or constraints. The use of KBS in music seems to be a natural choice especially when we try to model well defined domains or we want to introduce explicit structures or rules. Their main advantage is that they have explicit reasoning; they can explain their choice of actions.

Even though KBS seem to be the most suitable choice, as a stand alone method, for algorithmic composition they still exhibit some important problems:

- Knowledge elicitation is difficult and time consuming, especially in subjective domains such as music.
- Since they do what we program them to do they depend on the ability of the "expert",who in many cases is not the same as the programmer, to clarify concepts, or even find a fiexible representation.
- They become too complicated if we try to add all the "exceptions to the rule and their preconditions, something necessary in this domain.

8.2.3.3 Grammars

The idea that there is a grammar of music is probably as old as the idea of grammar itself.

Linguistics is an attempt to identify how language functions: what are the components, how do the components function as a single unit, and how do the components function as single entities within the context of the larger unit. Linguistic theory models this unconscious knowledge [of speech] by a formal system of principles or rules called a generative grammar, which describes (or 'generates') the possible sentences of the language. Curtis Roads has made a distinction between the specific use of generative grammars and the more open-ended field of algorithmic composition in that "Generative modeling of music can be distinguished from algorithmic composition on the basis of different goals. While algorithmic composition aims at an aesthetically satisfying new composition, generative modeling of music is a means of proposing and verifying a theory of an extant corpus of compositions or the competence that generated them."

Experiments in Musical Intelligence (EMI) is a project focused on the understanding of musical style and stylistic replication of various composers (Cope, 1991, 1992). EMI needs as an input a minimum of two works from which extracts "signatures" using pattern matching. The meaningful arrangement of these signatures in replicated works is accomplished through the use of an augmented transition network (ATN).

Some basic problems of the grammars are:

- They are hierarchical structures while much music is not (i.e improvisation). Therefore ambiguity might be necessary since it "can add to the representational power of a grammar".
- Most, if not all, musical grammar implementations do not make any strong claims about the semantics of the pieces.
- Usually a grammar can generate a large number musical strings of questionable quality.
- Parsing is, in many cases, computationally expensive especially if we try to cope with ambiguity.

8.2.3.4 Evolutionary methods

Genetic algorithms (GAs) have proven to be very efficient search methods, especially when dealing with problems with very large search spaces. This, coupled with their ability to provide multiple solutions, which is often what is needed in creative domains, makes them a good candidate for a search engine in a musical application. Taking inspiration from natural evolution to guide search of problem space, the idea is that good compositions, or composition systems can be evolved from an initial (often random) starting point.

Algorithm `GeneticAlg`

```
Initialise population
while not finished evolving
   Calculate fitness of each individual
   Select prefered individuals to be parents
   for N ¡= populationSize
      Breed new individuals
      (cross over + mutation)
```

Build next generation

Render output

We can divide these attempts into two categories based on the implementation of the fitness function.

Use of an objective fitness function. In this case the chromosomes are evaluated based on formally stated, computable functions. The efficacy of the GA approach depends heavily on the amount of knowledge the system possesses; even so GAs are not ideal for the simulation of human musical thought because their operation in no way simulates human behaviour.

Use of a human as a fitness function. Usually we refer to this type of GA as interactive-GA (IGA). In this case a human replaces the fitness function in order to evaluate the chromosomes.

These attempts exhibit two main drawbacks associated with all IGAs:

- Subjectivity
- Efficiency, the ”fitness bottleneck. The user must hear all the potential solutions in order to evaluate a population.

Moreover, this approach tells us little about the mental processes involved in music composition since all the reasoning is encoded inaccessibly in the users mind. Most of these approaches exhibit very simple representations in an attempt to decrease the search space, which in some cases compromises their output quality.

8.2.3.5 Systems which learn

In the category of learning systems are systems which, in general, do not have a priori knowledge (e.g. production rules, constraints) of the domain, but instead learn its features by examples. We can further classify these systems, based on the way they store the information, to subsymbolic/distributive (Artificial Neural Networks, ANN) and symbolic (Machine Learning, ML).

ANNs offer an alternative for algorithmic composition to the traditional symbolic AI methods, one which loosely resembles the activities in the human brain, but at the moment they do not seem to be as efficient or as practical, at least as a stand-alone approach. Some of their disadvantages are:

- Composition as compared with cognition is a much more highly intellectual process (more ”symbolic”).The output from a ANN matches the probability distribution of the sequence set to which it is exposed, something which is desirable, but on the other hand shows us its limit: ”While ANNs are capable of successfully capturing the surface structure of a melodic passage and produce new melodies on the basis of the thus acquired knowledge, they mostly fail to pick up the higher-level features of music, such as those related to phrasing or tonal functions”.
- The representation of time can not be dealt efficiently even with ANNs which have feedback. Usually they solve toy problems, with many simplifications, when compared with the knowledge based approaches.
- They can not even reproduce fully the training set and when they do this it might mean that they did not generalise.

- Even though it seems exciting that a system learns by examples this is not always the whole truth since the human in many cases needs to do the ”filtering” in order not to have in the training set examples which conflict.
- Usually, the researchers using ANNs say that their advantage over knowledge based approaches is that they can learn from examples things which can’t be represented symbolically using rules (i.e. the ”exceptions”).

8.2.3.6 Hybrid systems

Hybrid systems are ones which use a combination of AI techniques. In this section we discuss systems which combine evolutionary and connectionist methods, or symbolic and subsymbolic ones. The reason behind using hybrid systems, not only for musical applications, is very simple and logical. Since each AI method exhibits different strengths then we should adopt a ”postmodern attitude by combining them.

The main disadvantage of hybrid systems is that they are usually complicated, especially in the case of tightly-coupled or fully integrated models. The implementation, verification and validation is also time consuming.

8.2.4 Discussion

First there is usually no evaluation of the output by real experts (e.g., professional musicians) in most of the systems and second, the evaluation of the system (algorithm) is given relatively small consideration

Knowledge representation Two almost ubiquitous issues in AI are representation of knowledge and search method. From one point of view, our categorisation above, reflects the search method, which however, constrains the possible representations of knowledge. For example structures which are easily represented symbolically are often difficult to represent with a ANN.

In many AI systems, especially symbolic, the choice of the knowledge representation is an important factor in reducing the search space. For example Biles (1994) and Papadopoulos and Wiggins (1998) used a more abstract representation, representing the degrees of the scale rather than the absolute pitches. This reduced immensely the search space since the representation did not allow the generation of non-scale notes (they are considered dissonant) and the inter-key equivalence was abstracted out.

Most of the systems reviewed exhibit a single, fixed representation of the musical structures. Some, on the other hand, use multiple viewpoints which we believe simulate more closely human musical thinking.

Computational Creativity Probably the most difficult task is to incorporate in our systems the concept of creativity. This is difficult since we do not have a clear idea of what creativity is (Boden, 1996).

Some characteristics of computational creativity, which were proposed by Rowe and Partridge (1993) are:

- Knowledge representation is organised in such a way that the number of possible associations is maximised. A flexible knowledge representation scheme. Similarly Boden (1996) says that representation should allow to explore and transform the conceptual space.

- Tolerate ambiguity in representations.
- Allow multiple representations in order to avoid the problem of ”functional fixity”.
- The usefulness of new combinations should be assessable.

New combinations need to be elaboratable to find out their consequences. One question that AI researchers should aim to answer is: do we want to simulate human creativity itself or the results of it? (Is DEEP BLUE creative, even if it does not simulate the human mind?) This is more or less similar to the, subtle in many cases, distinction between cognitive modeling and knowledge engineering.

8.2.5 Emerging Trends

There are trends that, while being foreign to the Music Generation Modelling domain, propose issues that need to be taken in due consideration because they may condition the musical creation in the very near future[2].

Internet as a Participation Architecture The Internet is developing as an architecture of participation. There is a fast development of support to the creation of musical subcultures. In fact new musical styles develop with a speed never seen before. The spreading of new works of art happens through peer recommendation. In this way the Internet contributes to social innovation and the creation of social interaction and integration without much of geographic boundaries. Even language boundaries are less important in the musical domain, which stimulates the emergence of World Musics. In this way music and Internet have functions that create mutual synergy. In this way music can become an antidote for individualism. Technology could help in bringing people together through musical communication and interaction. However, many of these new systems depend on information gathering technologies that cannot stand the test of acceptable user privacy and on the other hand social participation and effects of entrainment are not well understood. What kind of participative technologies are needed in this domain?

Music as a Multi-Modal phenomenon Up to recent, most music technology researchers have associated music research with audio. Yet the above trend shows that music is in fact grounded on multi-modal perception and action. The way music is experienced in non-Western cultures and in the modern Wester popular culture is a good example of the multi-modal basis of music, e.g. its association with dance, costumes, decor etc... The multi-modal aspect of musical interaction draws on the idea that the sensory systems, auditory, visual, haptic, and tactile, as well as movement perception, form a fully integrated part of the way the human subject is involved with music during interactive musical communication. However, the multi-modal basis of the musical experience is very badly understood, as is the coupling between perception and action. A more thorough scientific understanding of the multi-modal basis of music, as well as of the close interaction between perception and action, is needed in view of the new trends towards multimedia.

Active Listener Participation Looking at the consumption pattern of people, there is also a trend which shows that people become more active consumers. For example, children nowadays like more their computer environment than television because they can be active with it. We don’t consume what is presented to use, but we perform actions and we choose. (Digital television is likely to focus on

[2] *adapted from Sound and Music Computing roadmap, $S2S^2$ project (in preparation)*

this new type of consumers in the near future in that it will offer programs with active participation.) In music creation and performance, active participation of the audience is likely to become a new trend provided that there is a technology which processes the actions of the consumer and feed them back into the performance. More research is needed in exploring technology as an extension of the human body, capture responses of the human body, as individual and as a group, and allow active participation of the participant. This involves massive wireless networking of many people gathered in indoor or outdoor theatres, which goes much beyond any present day mobile technology capacity density, and high quality portable music equipment.

8.3 Commented bibliography

A good tutorial on Hidden Markov Models is Rabiner [1989]. Hiller and Isaacson Hiller and Isaacson [1959] were the first to implement Markov chains in a musical application. The application of HMM representation of musical theme for search, described in Sect. 8.1.3.1, is presented in Shifrin et al. [2002].

References

Lejaren A. Hiller and L. M. Isaacson. *Experimental Music-Composition with an Electronic Computer*. McGraw-Hill, 1959.

L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceeedings of the IEEE*, 77(2):257–286, 1989.

J. Shifrin, B. Pardo, C. Meek, and W. Birmingham. Hmm-based musical query retrieval. In *Proc. ACM/IEEE Joint Conference on Digital Libraries*, pages 295–300, 2002.

Iannis Xenakis. *Formalized Music*. Indiana University Press, 1971.

Contents