Journal of Mathematics and Music
Vol. 00, No. 00, January 2008, 1–21

RESEARCH ARTICLE

Minimum Description Length Modeling of Musical Structure

Panayotis Mavromatis[a*]

[a]*Music and Audio Research Lab (MARL)*
and
Department of Music and Performing Arts,
New York University, 35 West 4th Street, Suite 777,
New York, NY 10012, USA

(*March 10, 2009*)

This article presents a method of inductive inference whose aim is to build formal quantitative models of musical structure. The models are constructed by statistical extraction of significant patterns from a musical corpus. The Minimum Description Length (MDL) principle is used to select the best model from among members of a non-parametric model family characterized by an unbounded parameter set. The chosen model achieves optimal compromise between goodness-of-fit and model complexity, thereby avoiding the over-fitting normally associated with such a family of models. The MDL method is illustrated through its application to the Hidden Markov Model framework. We derive an original mathematical expression for the MDL complexity of Hidden Markov Models (HMMs) that employ a finite alphabet of symbols; these models are particularly suited to the symbolic modeling of musical structure. As an illustration, we use the proposed HMM complexity expression to construct a model for a common melodic formula in Greek church chant. Such formulas are characterized by text-tune association that is governed by complex rules. We show that MDL-guided model construction gradually "learns" important aspects of the melodic formula's structure, and that the MDL principle terminates the process when nothing significant is left to learn. We outline how the musical applications of MDL can be extended beyond the HMM framework, and how they address general methodological concerns in empirical music research.

Keywords: computational modeling; model selection; inductive inference; information theory; pattern recognition; machine learning; Minimum Description Length; Hidden Markov Models; text-tune association; Greek chant

AMS Subject Classification: 68T05; 68T10; 94A17; 94A24; 60J20

1. Introduction

The problem of model selection is central to all empirical studies: given a set of observations, investigators seek to construct models that explain the data, offering insight into the underlying, and often hidden, mechanisms that account for the observed behavior. In the process of discovery, the researcher is faced with the task of choosing among competing models, carefully evaluating each of them using a number of different, and sometimes conflicting, criteria.

One important line of music research aims to investigate musical structure from an empirical perspective. In one empirical approach, researchers formulate questions quantitatively, and subsequently answer them by analyzing patterns in a body

*Corresponding author. Email: panos.mavromatis@nyu.edu

ISSN: 1745-9737 print/ISSN 1745-9745 online

DOI: 10.1080/1745973YYxxxxxxxx
http://www.informaworld.com

of music data, often employing statistical tools [see, e.g., 1]. In another approach, researchers formulate rule systems that help *explain* the patterns in the data; they subsequently evaluate and refine these rules based on their accuracy, predictive power and other criteria [2]. In both these approaches, model selection plays an important role.

This article offers a contribution to the above line of empirical music research; its purpose is to present a general method for constructing and evaluating quantitative models of musical structure. These models capture the detailed behavior of musical objects characterizing a given style or practice, such as phrases, melodic formulas, and ultimately complete compositions. We show how the information necessary to build such models can be extracted from a corpus of examples representative of the style in question. This derivation of general rules from specific examples is an instance of *inductive inference*. The *Minimum Description Length* principle is a powerful general technique of inductive inference that evaluates quantitative models and guides their construction [3, 4].

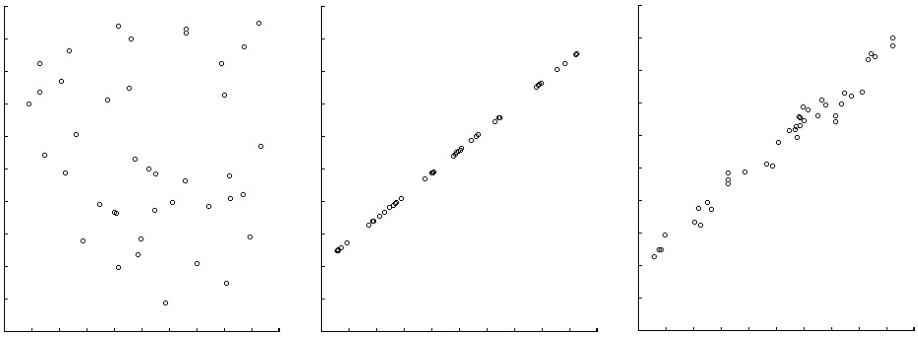
In formal modeling, musical structure is characterized by symbolic variables representing pitch, duration, etc. It may also be desirable to include variables for relevant non-musical structure, such as words set to music. A model's purpose is to identify syntactical constraints on the values of these variables. Such constraints are manifested as statistical regularities in the occurrence of symbol combinations, which create meaningful recurring patterns of behavior. Accordingly, it is desirable to have a method for detecting and extracting significant regularities from the corpus under investigation. These regularities represent stylistic norms, and may offer insight into the mechanisms that generated the corpus, such as internalized knowledge of composers proficient in the style.

The search for meaningful patterns in data represents a particular type of inductive inference known as *pattern recognition*; the latter is a major area of focus in machine learning research [5]. Pattern recognition problems are typically governed by complex relations that resist modeling through "hand-crafted" rules; an algorithmic approach to model building is therefore necessary. An additional advantage to this algorithmic approach is that it can effortlessly handle large quantities of data. In addition, by being formally specified, algorithmic model-building minimizes any arbitrary or ad-hoc aspects in model selection, thereby placing the process on a sound methodological basis.

In quantitative modeling, a good fit between the model and the data is obviously always desirable. Accordingly, *goodness-of-fit* is a central focus of statistical techniques, and can be readily quantified in different ways. A more elusive and less obviously quantifiable modeling goal is to avoid *over-fitting*; the latter refers to a situation where a model fits a given data set too well, capturing its insignificant idiosyncrasies, and failing to achieve the right level of abstraction and generalization. Over-fitting has been the focus of much research, and many ad hoc approaches have been proposed to address it.

One of the main strengths of the Minimum Description Length (MDL) principle is that it offers a powerful general solution to the problem of over-fitting. Inspired by information theory and data compression, MDL evaluates a candidate model according to (i) how well it fits the data, and (ii) how complex it is, seeking an optimal compromise between these two opposing tendencies. Since unnecessary model complexity lies at the heart of over-fitting, the latter is kept in check in a principled and elegant way. The resulting model selection framework has a number of desirable properties, and finds application in many classic problems of statistical inference and machine learning [3, 4].

The purpose of this article is to present MDL model selection in the context



(a) (b) (c)

Figure 1. (a) A general data set consisting of N points on a two-dimensional plane. The data points show no identifiable regular pattern, and so $2N$ real parameters are needed to communicate the data set. (b) A data set consisting of N points on a two-dimensional plane that obey a *deterministic* linear law $y = ax + b$. Only $N + 2$ real parameters are now needed to communicate the data. Two of these parameters characterize the model (the straight line); the remaining N parameters characterize the data *given* the model. Therefore, the underlying regularity permits considerable compression of the data. (c) A data set consisting of N points on a two-dimensional plane that obey a *probabilistic* linear law $y = ax + b$. The compression possible for this data set is in between those for (a) and (b), and depends on the "errors," or deviations from the exact linear relation.

of formal computational approaches to music; moreover, to demonstrate the principle at work in the Hidden Markov Model framework, a formalism particularly suited to the modeling of musical structure [6, 7]. In particular, we will present an original mathematical expression for the MDL complexity of Hidden Markov Models (HMMs) that employ a finite alphabet of symbols. With the help of the proposed expression, this class of models can be used to tackle a variety of musical applications [8].

As an illustration, we will use the HMM complexity expression to construct a model for a common melodic formula in Greek church chant. Such formulas are characterized by text-tune association that is governed by complex rules. We will show that MDL-guided model construction gradually "learns" important aspects of the melodic formula's structure, and that the MDL principle terminates the process when nothing significant is left to learn.

This article is organized as follows: in Section 2, we will present an overview of the MDL model selection framework, highlighting the relation between regularity and data compression. We will also show how MDL can control over-fitting by quantifying model complexity. In Section 3, we will outline the HMM framework and show how over-fitting can arise in this context. In Section 4, we will illustrate these ideas with the help of the Greek chant modeling application. In Section 5, we will discuss the broader implications of our MDL approach for modeling musical structure. In particular, we will outline how the musical applications of MDL can be extended beyond the HMM framework, and how they address general methodological concerns in empirical music research.

2. MDL Model Selection: An Overview

2.1. Regularity and Data Compression

The fundamental idea behind the Minimum Description Length (MDL) principle is that any regularity identified in a data set can be used to *compress* that data set. Here to "compress" means to encode the data in a form much shorter than the form involved in a literal description. Before we formalize this idea, let us illustrate the intuition behind it with a simple, albeit somewhat crude, example.

Consider a data set consisting of points in a two-dimensional plane, such as the one shown in Figure 1(a). We wish to encode the data so as to be able to communicate them over a channel, such as a computer network. For that purpose, each data point can be represented by two real numbers, namely the coordinates of that point on the plane. Assuming there are N points in the data set, transmitting $2N$ real coordinates is sufficient for reconstructing the data set at the other end of the channel. Since nothing more is assumed of the data set, the length $2N$ corresponds to a "literal" description of the data.

The above situation represents, in fact, a worst-case scenario. Consider, for example, what happens if all the data points fall on a straight line, as shown in Figure 1(b). In this case, a linear relation $y = ax + b$ can be assumed between the coordinates (x, y) of each data point, where a and b are fixed real numbers. This relation captures a regularity in the data set, and can be used to shorten the description of the data. Indeed, given the linear coefficients a and b, we can reconstruct any given data point using a *single* real parameter, say its coordinate x, from which its y coordinate can also be calculated. To characterize the data set completely, we therefore need N real parameters for the data points, plus two real parameters for the linear relation. This amounts to a total description length of $N + 2$, which is in general much shorter than $2N$.

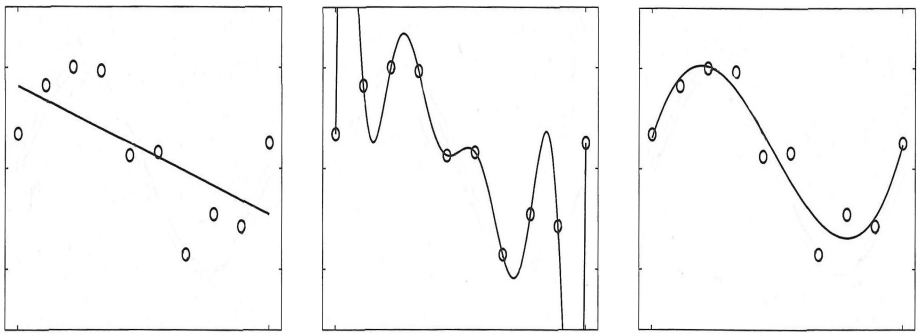
Figure 1(b) represents a situation that is admittedly somewhat idealized. Empirical observations typically yield data sets that look more like Figure 1(c). In the latter case, an overall *approximate* linear relation can be discerned for the data set as a whole. However, as far as individual data points go, it is simply expected that they will fall somewhere *near* the ideal straight line; their exact location will be governed by a *probability distribution*. Here we are dealing with a regularity that is *statistical* rather than *deterministic* in nature. How can a regularity of this type be exploited to achieve a shortened description of the data?

Intuitively, the argument goes as follows: as in the deterministic case, we begin by transmitting the $N+2$ real numbers representing the two linear coefficients plus the *idealized* position of each data point on the straight line—i.e., the position on the line that is closest to the actual data point. However, this information must now be supplemented by the "errors," or perpendicular distances between the idealized and actual data points. In the worst case, this would amount to another set of N real parameters, which is no better than the situation of Figure 1(a). As it turns out, however, we can exploit the probability distribution governing the errors to *recode* the latter in such a way as to assign *shortest code lengths to the most likely* (i.e. smaller) *errors and longer code lengths to the less likely* (i.e. longer) *ones*. This is one of the central insights offered by the MDL framework, and will be made precise in Section 2.3. For now, suffice to say that the distribution-aware encoding of the errors involves description length L_{err} shorter than the worst-case length N. Consequently, the data set of Figure 1(c) admits compression that is in-between the compressions possible for Figures 1(a) and 1(b), as quantified in the relation

$$N + 2 < N + 2 + L_{err} < 2N$$

Moreover, it can be shown that the closer the linear relation fits the data—i. e., the smaller the errors involved—the better the compression achieved through this method.

Whether statistical or deterministic, regularities such as the linear relations identified above constitute *laws* that govern the behavior of the data. Establishing such laws is of central concern to inductive inference and forms an integral part of model-



(a) (b) (c)

Figure 2. (a) A polynomial curve (straight line) that is too coarse to capture the underlying regularities of a data set, exemplifying under-fitting. (b) A polynomial curve that is too complex and hence over-fits the data set. (c) A polynomial curve displaying optimal balance between goodness-of-fit and model complexity; such a curve typically enjoys maximum explanatory and predictive power.

building. The MDL principle asserts that the more we are able to compress a data set, the more we have *learned* about it. This knowledge, in the form of laws, offers insight into the mechanisms that generated the data, and leads to models that can claim considerable *explanatory power*.

But are the above modeling ideals always achieved by imposing more and more relations on a data set? Does a complex model always achieve better compression than a simple one? As it turns out, this is not necessarily the case.

2.2. The Problem of Over-Fitting

One of the most important concerns in model selection is the problem of *over-fitting*. The term is used to describe a situation where an overly complex model fits the observed data very closely but predicts future data very poorly.

To illustrate how this situation can arise, consider the classic example of *polynomial curve-fitting*, also known as *polynomial regression*. In this application, a two-dimensional data set is assumed to obey a statistical law of the form

$$y = \sum_{k=0}^{n} a_k x^k + err(x) \tag{1}$$

for suitable values of the polynomial degree n and coefficients a_k. As in Figure 1(c), the deviation of the data from a deterministic law is represented by an error term $err(x)$, usually assumed to obey a normal distribution. The goal is to find the polynomial that best models the behavior of the data set. This involves determining the polynomial's degree as well as its coefficients.

For a *given* degree n, the standard procedure for determining the polynomial coefficients a_k is by minimizing the squared error

$$\sum_{i=1}^{N} \big[err(x_i)\big]^2 = \sum_{i=1}^{N} \left[y_i - \sum_{k=0}^{n} a_k (x_i)^k\right]^2 \tag{2}$$

where the summation is over all data points. This *goodness-of-fit* criterion works well when n is known in advance. However, if n needs to be determined, goodness-of-fit by itself can lead to problems.

Consider the data set represented by the points in Figure 2. Each of the curves in 2(a)–2(c) represents three different situations that may arise when fitting a

polynomial of a particular degree to the data set.

Polynomial 2(a)—a straight line—is too simple to capture the overall up-and-down shape of the data set, which is seemingly one of its essential regularities. This is reflected in the large distances of the data points from the curve, which translates into higher squared error, and therefore poor goodness-of-fit. The source of the problem is that a linear relation involves *too few parameters*, and so the parameter space is too small to minimize the squared error in a satisfactory way. We will refer to this situation as *under-fitting*.

By contrast, polynomial 2(b) appears to err in the opposite direction. In this case, there are *too many parameters*; this leads to a very small squared error, and hence a very tight fit to the data. In fact, a polynomial of degree $N-1$ can go through *any* set of N data points with zero squared error, achieving perfect fit. However, the contour of such a high-degree polynomial seems overly complex; its up-and-down shape appears to capture not only the essential regularities in the data set but also the random fluctuations, or *noise*, of its probabilistic source. This situation is known as *over-fitting*.

Polynomial 2(c) represents a happy middle-ground between the above two extremes. It captures the desired *overall* behavior of the data set, but does not fall prey to the noise contained in it. It is precisely this behavior that the optimal model is expected to display.

In fact, both under-fitting and over-fitting lead to models with poor *predictive power*, since in both cases the structure of the model does not reflect the structure of the data source. In the former case, new data may not conform to the over-simplified shape of the identified curve; in the latter case, they may not conform to the curve's random aspects. This problem is not limited to polynomial curve-fitting, but characterizes more generally *all families of models that involve an unbounded number of parameters*. One should exploit these parameters to optimize fit; at the same time, one should not over-indulge in this freedom, or over-fitting will occur.

One of the special strengths of the MDL principle is that it offers an elegant general solution to the above model-selection conundrum. We have already shown informally that MDL encourages goodness-of-fit through the proper encoding of the data point errors, or deviations from the ideal curve. But how does that principle protect from over-fitting?

Recall that the MDL principle seeks to identify the model that compresses the data the most. In Section 2.1 we saw that this compression is achieved by (i) encoding the data set with the help of a model, and (ii) encoding the model itself as a preamble to the data description; this second step is necessary for the data to be unambiguously decodable. From this it follows that the *complexity* of a model, as reflected in its parameter set, may well have an *adverse* effect on data compression: any gain achieved through that model's improved fit may, in fact, be offset by the additional length required to describe the model itself. Since, as we saw, over-fitting is a direct consequence of model complexity, a proper quantification of the latter should provide the appropriate term to counterbalance goodness-of-fit. The MDL principle will then select the model that offers the most compact description of the data by achieving *optimal balance between model complexity and goodness-of-fit*.

The description lengths used in our informal illustrations so far do not achieve optimal compression, and hence are too crude to show MDL model selection at work. Therefore, the intuitive discussion of Section 2.1 must be cast in precise quantitative form. This question will be addressed next.

2.3. Formalizing MDL: The Two-Part Code

To formalize the MDL principle, we must specify what we mean by "encoding" the data and the model. Once a coding scheme C is formally defined, we can associate with it a *code-length function* L_C that quantifies the description length achieved through it.

The simplest formalization of the MDL principle is known as the *two-part code*. In this formulation, the description length $L_{total}(D, M)$ to be minimized is defined as the sum of two parts:

$$L_{total}(D, M) = L_{C_1}(M) + L_{C_2}(D|M) \tag{3}$$

As suggested in Section 2.1, the first term $L_{C_1}(M)$ represents the code length for the candidate model, and the second term $L_{C_2}(D|M)$ represents the code length for the data set as encoded with the help of that model. Formalization is achieved by specifying the *coding schemes* C_1 and C_2 appropriate for the model and data respectively. In the remainder of this section we will outline how these coding schemes can be defined, highlighting the main concepts involved. The mathematical details of this formalization are presented in this article's Online Supplement, Sections 1 and 3.

We will assume that both the model and the data can be represented by sets of numeric or symbolic values, such as real numbers or—in the case of musical variables—note pitch, duration, etc. To be able to measure and compare description lengths, we need to place all these values on an equal footing. To that end, we will assume that all our coding schemes map these different representations into *strings of binary digits*.

Let the S stand for the set of objects to be encoded. In the case of data coding, S represents the sample space from which the data are drawn. Similarly, when coding a model, S is the space of all models under consideration, such as the set of all polynomials of arbitrary degree. Let S^* denote the set of all finite sequences of symbols in S.

A *coding system* on S^* is defined by a function $C : S \to \mathcal{B}$ from S to the set $\mathcal{B}$ of binary strings of arbitrary length. For $s \in S$, we say $C(s)$ is a *codeword* for s. Furthermore, we extend C to encode sequences of symbols $s_1 s_2 \dots s_n \in S^*$ by mapping each sequence to the concatenation of the individual symbols' codewords. In other words, $C(s_1 s_2 \dots s_n) \equiv C(s_1)C(s_2)\dots C(s_n)$. The *code-length function* $L_C : S^* \to \mathbb{Z}^+$ of C assigns to each sequence $s_1, s_2, \dots s_n \in S^*$ the length of the binary string $C(s_1 s_2 \dots s_n)$ that encodes the sequence under C.

The MDL principle assumes that no information is lost in the encoding. Accordingly, for a coding system to be useful for our purposes, one must be able to *decode* a sequence of codewords to retrieve the sequence of objects that it represents. In other words, the coding system must be an *invertible function* from S^* to $\mathcal{B}$. In the MDL framework, this is accomplished by choosing C to be a *prefix code*.

More specifically, let b_1, b_2 be binary strings; b_2 is a *prefix* of b_1 if $b_1 = b_2 b'$ for some binary string b'. We say a coding system $C : S^* \to \mathcal{B}$ is a *prefix code* provided that (i) C is a one-to-one function, and (ii) there are no two codewords $C(s_1), C(s_2)$ such that $C(s_1)$ is a prefix for $C(s_2)$. The motivation behind the use of prefix codes is that they allow us to determine unambiguously where each codeword ends and where a new one begins, without the use of a special symbol to mark word boundaries.

The above framework allows us to calculate the code-length functions $L_{C_1}(M)$ and $L_{C_2}(D|M)$ required by the two-part code of eq. (3). Let us first turn our attention to the data code-length function $L_{C_2}(D|M)$.

Given a model M, we wish to encode the data set D with the help of M in the most compact way. It turns out that there is a general solution to the problem that does not depend on the model's detailed structure. More specifically, the function $L_{C_2}(D|M)$ can be simply expressed in terms of the probability distribution defined by the model M.

The intuition behind this idea is that a maximally compact encoding should assign the *shortest* codewords to the *most likely* data symbols. Let $P(D|M)$ be the probability distribution defined by model M on possible data D. It can be shown that there is always an optimal choice C_P for the data coding system, in the sense that it minimizes the *expected* code length given the probability distribution $P(D|M)$. This coding system is known as the *Shannon-Fano code*, and has code-length function given by

$$L_P(D|M) = -\log_2 P(D|M) \tag{4}$$

Moreover, it can be shown that C_P is a prefix code, and is hence invertible, as required by the MDL principle.

We can develop a sense for the behavior of the Shannon-Fano code length function $L_P(D|M)$ by considering its expectation under the underlying probability distribution $P(x|M)$ for a single data point x:

$$H_M \equiv \sum_{x \in X} [-\log P(x|M)] P(x|M) \tag{5}$$

The quantity H_M is the well-known *entropy* of the probability distribution $P(x|M)$, one of the most important concepts in information theory. This entropy is commonly interpreted as a measure of randomness associated with the probabilistic data source represented by model M. In relation to Figure 1(a)–1(c), it would appear that source 1(a) has the highest entropy of the three, and that source 1(b) has the lowest. Thus, with the help of the entropy, it can be shown formally that the Shannon-Fano code is expected to assign data set 1(a) the longest code length, while 1(b) is expected to receive the shortest. This formalizes the intuitive discussion of data coding that was presented in Section 2.1.

We next turn to the model code-length function $L_{C_1}(M)$ in the two-part code of eq. (3). As it turns out, the coding system used to encode M depends on the model family to which M belongs, and therefore so does the form of the code-length function. Hence, the process of deriving $L_{C_1}(M)$ is best illustrated in the context of a specific model family. To that end, detailed treatment of this term will take place in the framework of Hidden Markov Models (HMMs). An original formula for the HMM code length function will be given in Section 3 (eq. 6), after briefly presenting the required background in the HMM formalism. The full derivation of (6) will be given in the Online Supplement, Section 3. With the help of this example, it will be apparent that the model encoding will have to capture both the model's *structure* and its *parameter set*. As a result, the more complex the model, as reflected in its structure and parameters, the larger the code length $L_{C_1}(M)$ assigned to it.

We are now in a position to show formally how the MDL criterion addresses overfitting, fleshing out the informal argument of Section 2.2. We have just argued that the model code-lenght function $L_{C_1}(M)$ quantifies *model complexity*. In fact, it can be shown that the data code length function $L_{C_2}(D|M)$ defined by eq. (4) is a natural measure of *goodness-of-fit*. The argument is detailed in the Online Supplement, Section 1.3. There, we show that the expression $-\log_2 P(D|M)$ for the

Shannon-Fano code of eq. (4) reduces to the more common measure of least-square errors (eq. 2) under the conditions in which the latter is commonly employed.

It is now clear that minimizing the second term in eq. (3) amounts to improving goodness-of-fit. Moreover, minimizing the first term in eq. (3) translates into improving model simplicity. Therefore, minimizing the *total* code length in eq. (3) can be interpreted as seeking the optimal compromise between goodness-of-fit and model simplicity. This is the recipe against over-fitting that was outlined at the end of Section 2.2. Equation (3) shows how the MDL criterion casts this recipe in precise quantitative form.

3. Hidden Markov Models

The Hidden Markov Model framework will be used in this article to illustrate MDL model selection. Hidden Markov Models (HMMs) offer a powerful flexible method of inductive inference that is used to model structure in sequential data [9–11]. A HMM is a probabilistic version of a *Finite State Machine*. The latter is a well-studied type of formal grammar that is equipped with a wide array of computational tools [12]. Finite State Machines represent *Type III grammars* in Chomsky's classification of formal languages [13], and are designed to model local non-hierarchical structure. Because of their ability to model a wide variety of syntactical relations, Finite State Machines assume central role in the definition and implementation of programming languages [14]. Their probabilistic extension, HMMs, have found application in many different disciplines, including linguistics, bioinformatics, and financial forecasting. More recently, HMMs have featured in music information retrieval [see, e.g., 15], as well as music-theoretic computational studies [16–19]. Their generality and expressive power makes HMMs a powerful tool for the study of musical structure [6, 8].

In this section, we will lay out the basic definitions of HMMs and formulate the problem of model selection in that framework. The presentation will remain informal, and more technical details will be presented in the Online Supplement, Section 2. We will limit our discussion to HMMs defined over a finite alphabet of symbols, as these are most relevant for modeling musical structure. The formalism will not be discussed beyond what is necessary for the present argument. In the context of music theory, HMMs have already been informally reviewed in [6, 7], to which the reader is referred for further information.

3.1. The Basic HMM Formalism

We will assume that the musical objects to be modeled are represented by symbolic sequence data, which we will refer to as the *observation sequence*. An observation sequence is represented as a set of variables $\{O_t : t = 1, 2, \ldots N\}$ taking values in a finite set of symbols $\mathcal{O} = \{o_k | k = 0, 1, \ldots K\}$ called the *output alphabet*. The goal of a formal grammar is to capture syntactical constraints among the successive values of the observation sequence $\{O_t\}$.

A *Hidden Markov Model* (HMM) is a simple, yet powerful probabilistic formal grammar. Its definition is outlined here, along with the commonly associated graphic representation. A HMM consists of the following:

- A set of *states* $\mathcal{S} = \{s_i\}$, graphically represented by circles, called *nodes*.
- A set of *state transitions* $\{(s_i, s_j)\} \subset \mathcal{S} \times \mathcal{S}$. The transition (s_i, s_j) is graphically represented by an arrow from state s_i to state s_i, called an *arc*.
- For each state transition (s_i, s_j), a set of *output symbols* $\{o_k\} \subset \mathcal{O}$ taken from

the output alphabet. Each output symbol is graphically shown next to its corresponding arc.

- For each state s_i, a set of *transition probabilities* a_{ij}, one for each transition (s_i, s_j), such that $\sum_j a_{ij} = 1$.
- For each state transition (s_i, s_j), a set of *output probabilities* b_{ijk}, one for each output symbol o_k on (s_i, s_j), such that $\sum_k b_{ijk} = 1$.

It is also customary to specify a HMM's *initial* and *final* states. For simplicity, in this paper we will designate a single state s_0 to act as both the initial and the final state. An example of a HMM is shown in Figure 4.

A *path* through a HMM is a sequence of states $\{S(t) : t = 0, 1, \ldots n\}$, where $S(0) = s_i$ is an initial state, $S(n) = s_j$ is a final state, and each $(S(t-1), S(t))$ is a state transition in the HMM. Let $\{O(t) : t = 1, \ldots n\}$ be a sequence of output symbols such that each $O(t) = o_k$ appears on the corresponding state transition $(S(t-1), S(t))$; we say that the output sequence is *generated* by the path $\{S(t)\}$. A sequence is judged by the HMM to be grammatical if and only if that sequence is generated by *some* path in the HMM. A given sequence may in fact be generated by more than one path, in which case it affords more than one grammatical interpretation in the model.

In addition to identifying all grammatical observation sequences, a probabilistic grammar also assigns a probability to each of them. This probability reflects how likely it is to encounter a given sequence. This feature is particularly pertinent to music modeling, where it may be desirable to characterize quantitatively which structures are typical and which are exceptional. In the MDL framework, this calculation is essential, as it supplies the Shannon-fano code length in equation (4).

In the case of HMMs, the probability of a sequence being generated by a *given* path can be simply calculated as the product of all transition and output probabilities encountered along the path. The *overall* probability assigned to the sequence by the HMM is then calculated as the sum of all probabilities coming from each path that generates the sequence.

3.2. HMM Inference from Data

Given a data set of observation sequences, the fundamental problem of HMM inference is to identify the HMM that best models the data set. This model selection problem involves determination of the following components:

(i) the number of states n_S;
(ii) the set of state transitions;
(iii) the output symbol(s) on each state transition;
(iv) the transition probabilities a_{ij}; and
(v) the output probabilities b_{ijk}.

Components (i)–(iii) define the HMM's *topology*; components (iv)–(v) are referred to as the HMM's *model parameters*.

The size of a HMM's parameter set is directly related to the model's number of states n_S. To see this, let n_A be the number of distinct alphabet symbols that make up the observation sequences. Then a HMM with n_S states can have up to n_S^2 state transitions and up to n_A output symbols on each transition, each of them carrying a probability value. It is therefore clear that HMMs form a model family that is characterized by an unbounded parameter set. As is the case with all such families, we expect that over-fitting is a central issue in HMM inference.

We have already seen how the MDL criterion solves the over-fitting problem by selecting the model which minimizes eq. (3), thereby negotiating a balance between

goodness-of-fit and model complexity. In Section 2.3, we saw how to calculate the data code-length term $L_{C_2}(D|M)$ in the two-part code of eq. (3). As it turns out, the model code-length function $L_{C_1}(M)$ for HMMs is given by the following formula:

$$L_{HMM}(M,d) = L^*(n_A) + L^*(n_S) + L^*(d) + (n_S+1)\log_2 \frac{(d+n_S+1)!}{d!\ n_S!} + n_T \log_2 \frac{(d+n_A+1)!}{d!\ n_A!} \tag{6}$$

where $L^*(n)$ is a real-valued function known as the *universal prior for non-negative integers* and d is an auxiliary parameter used to encode the HMM parameters. The derivation of this expression is detailed in the Online Supplement, Section 3, where all the relevant quantities are fully explained.

With the help of the Shannon-Fano code length (4) and the HMM complexity formula (6), it is straightforward to evaluate a *given* HMM according to the MDL criterion. However, one question still remains: given the enormous space of all possible HMMs, how can we search effectively for the best candidate models without going explicitly through all the possibilities? This is a serious problem which, if not properly addressed, threatens to render the technique unusable in practice.

For a better understanding of HMM model selection, it is instructive to note its similarity with the polynomial curve-fitting problem, which was used as an example in Section 2.2. There, we noted that it is easier to first solve the subproblem in which the polynomial's degree is specified. If this is the case, the number of model parameters—in that case, the polynomial coefficients—is fixed, and so we can estimate the parameter values using goodness-of-fit. Once we have identified the best polynomial of each degree, we can compare the winning polynomials against each other in order to determine which of them achieves reasonable fit without over-fitting the data.

In the case of HMM inference, we can proceed in an analogous way by first solving the model selection problem on a *fixed parameter set*. For HMMs, this is achieved by fixing the model's *topology*. Once more, the model parameters will be determined by optimizing goodness-of-fit, which in our case is represented by the data description length. After identifying the best parameter set for each topology, we can compare the corresponding HMMs, in order to choose the model that minimizes the *total* description length.

The details of the search over the space of HMMs are beyond the scope of this article. Suffice to say that the determination of HMM parameters for fixed topology can be achieved through the *Baum-Welch algorithm* [9, 10]. The latter maximizes the term $P(D|M)$, which has the effect of minimizing the data description length. The search over model topologies has been explored in a number of works [6, 8, 20–22]. In Section 4, we will present a simple example of HMM inference that illustrates topology determination by successive *state splitting*.

4. Example: Structure of a Melodic Formula in Greek Church Chant

As discussed earlier, HMMs are well-suited for modeling a variety of local non-hierarchical syntactical relations. Nevertheless, for the sake of concrete illustration it will be best to now turn to a specific example. To that end, this section will present the MDL principle at work by applying it to the problem of *text-tune association*. More specifically, our goal will be to model the structure of a melodic formula in Greek church chant. The shape of such formulas is governed by relations

between melodic pitch and word stress. The MDL principle will be used to construct a HMM which is a *generative text-setting model* of the formula in question. In other words, the model will map short fragments of text to their corresponding formulaic realization. The melodic idiom of Greek church chant was studied more extensively in [6, 7], to which the reader is referred for more details.

This simple example of model selection illustrates how MDL can be used to construct increasingly more refined HMMs from the data, and how each stage in the process captures an important aspect of the musical structure under investigation. The example also illustrates how this process of refinement terminates before overfitting sets in.

4.1. Text-Tune Association and Formulaic Chant Structure

Text-tune association has attracted the attention of music theorists and linguists alike [23–26]. The goal of this research is to discover formal rules that govern the setting of words to music. As many studies have shown, word stress plays an important role in determining the placement of text with respect to the contour and rhythmic profile of a melodic line. For instance, stressed syllables have the general tendency to fall on stronger beats than do unstressed syllables. The detailed structure of the text-tune association rules largely depends on the musical style, as well as the prosodic properties of the language in question. Failure to conform to such rules typically results in diminished intelligibility of the sung text.

A recent study by Mavromatis [6, 7] explored text-tune association in Greek church chant, a melodic idiom where oral transmission and improvisation play an important role. Using HMM analysis, Mavromatis obtained a generative model of text-setting that captures the pitch-stress constraints of the style. In particular, the model demonstrates the central importance of *melodic formulas* for the structure of Greek chant melodies. These formulas are recurring melodic segments six to nine notes long that embody the pitch-stress constraints, and that can be flexibly adjusted to accommodate a variety of word-stress patterns. Melodic formulas figure prominently at the end of phrases, where they assume cadential role.

To understand how word stress influences the shaping of Greek chant melody, it will be helpful to examine the structure of a particular melodic formula. To that end, let us consider one of the most commonly employed cadential formulas, several instances of which are listed in Figure 3. In these examples, text setting is understood to be predominantly syllabic, with a slur marking the occasional setting of two notes to a syllable. Each staff line corresponds to one of the formula's major variants, labeled (a1)–(c2). The cause of this variation is the changing pattern of word stress in the underlying text. The stress pattern corresponding to each variant is shown under its staff in grid notation [27], where two `x`'s represent a stressed syllable and one `x` represents an unstressed syllable. Parentheses are used to mark minor variations within the basic realization represented by each line.

As the examples of Figure 3 suggest, a melodic formula is an abstract schema that can be realized in different ways according to text-setting needs. At first glance, the schema for this particular formula seems simple enough; it involves a basic pitch sequence $< G, F, E, F, G >$ that can be adapted by insertions and deletions of notes. However, closer examination reveals a wide range of variation in pitch-stress mapping among the different variants (a1)–(c2). This variation presents a challenge to the task of building a precise formal characterization of the formula's structure. And yet despite this complex variability, all the realizations appear to share a formulaic identity that is clearly perceived by carriers of the musical idiom.

To better appreciate the complexity of the modeling task at hand, it may be

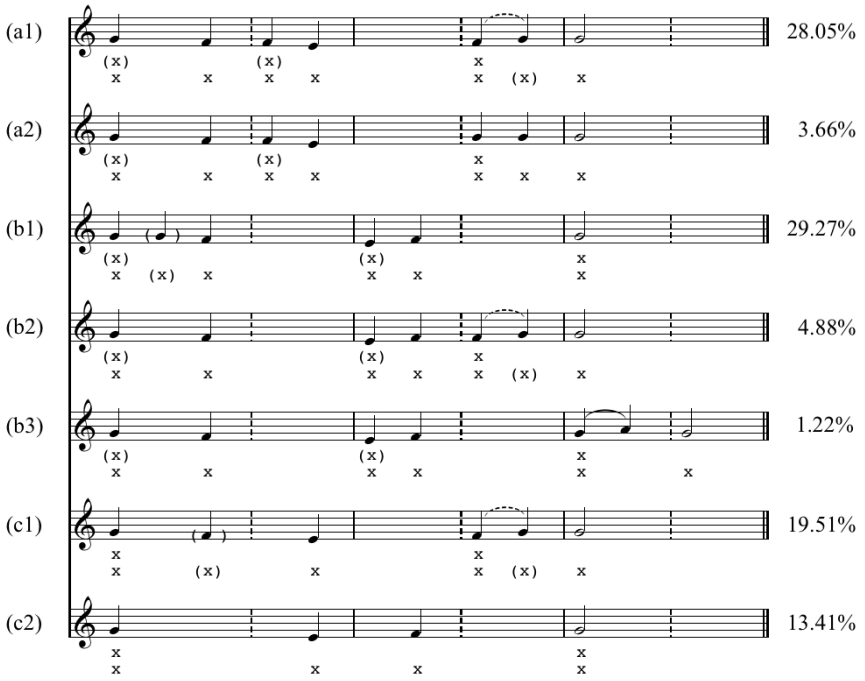


Figure 3. Instances of a cadential formula commonly employed in Greek church chant. On the right of each staff appears the relative frequency of occurrence for the corresponding variant. This percentage is based on the sample used for the HMM inference of Section 4.2.

instructive to attempt some formal rules that will explain, as well as delimit, the variability in our formula's realizations. Let us begin with a few observations in the form of rule-like generalizations:

(R1) The first G of the formula generally falls on a stressed syllable.
(R2) If the final syllable is stressed, then it is assigned to the final G. If, on the other hand, the last stressed syllable is followed by unstressed syllables, then it is generally assigned to the penultimate F; in that case, the unstressed syllables that follow it are assigned to the pitch G.
(R3) At most one stressed syllable can intervene between the initial G and the last stressed syllable of the formula; if present, this intermediate stress is assigned to either an F or to an E.
(R4) Once the stressed syllables are fixed according to rules (R1)–(R3) above, the variable number of unstressed syllables falling in-between them can be accommodated by appropriate insertions and deletions of notes.

Rules (R1)–(R4) are certainly insightful, in that they identify some key pitch-stress associations that shape the formula. In doing so, they offer a useful starting point for modeling. However, as they stand, these rules are not only vague—in (R3) above, how do we choose between an F and an E?—but they are also limited in scope or detail—the first G of the formula *can* occasionally fall on an unstressed syllable, contrary to the general guideline of (R1). In other words, the rules will have to be elaborated with qualifications and exceptions.

It is clear that, when approached in this way, the modeling task quickly becomes unmanageably complex. This complexity is seemingly necessary in order to account for the formula's flexible adaptability to a wide range of stress patterns. At the same time, the details of the complex rule formulation are important, since they provide the appropriate limits to the possible variations within the formulaic environment. This delimitation is key to preserving the formula's essential features, and hence its unmistakable identity and clear function as cadential punctuation. Simply put,

our model should neither *under-generalize* nor *over-generalize*.

To summarize, our first attempt at modeling tune-text association teaches us a couple of important lessons. First, the underlying rules may turn out to be too complex to derive "by hand" efficiently and reliably; an algorithmic model building technique is therefore desirable. Second, when faced with a complex array of possibilities, model evaluation has to be precise. In this way, the selected model will capture exactly the right amount of information, and will be neither too strict nor too relaxed. Therefore, a principled and precise model selection method is called for. To that end, we will reconsider this section's modeling problem in the MDL framework.

4.2. Building a HMM of Formulaic Structure in the MDL Framework

The pitch-stress relations that shape our cadential formula appear to be local and non-hierarchical in nature. This suggest that the rules we seek fall within the expressive power of a Finite State Machine grammar. Therefore, HMMs are a natural choice of formalism to approach our modeling problem.

As we saw in Section 3.2, inference of HMM structure is accomplished with the help of a data set that exemplifies the behavior to be modeled. For that purpose, we collected a random sample of 246 realizations of our cadential formula, all of them taken from prominent printed sources of modern Greek church chant [28]. All realizations belonged to one of the seven categories (a1)–(c2) identified in Figure 3. The relative frequency of occurrence for each category appears on the right of the corresponding staff in Figure 3.

Since we are interested in pitch-stress relations, our observation sequence must contain both word stress and pitch information in some appropriate symbolic form. To represent word stress, we will use a variable W_t taking three possible values $\{\mathrm{xx}, \mathrm{x\_}, \mathrm{\_\_}\}$, corresponding to *stressed syllable*, *unstressed syllable*, and *no syllable change* respectively. Pitch will be represented by a variable P_t receiving the four possible values $\{E, F, G, A\}$; these are simply the letter names of the pitches occurring in our cadential formula (see Figure 3). In addition to absolute pitch information, it is useful to explicitly include melodic intervals in our symbolic representation. To that end, we will use a variable I_t to represent diatonic intervals as integers. For instance, a step up will be represented by '1', and a third down by '-2'; likewise, a unison will be represented by '0'.

Concurrent pitch-stress combinations will be represented by the composite observation variable $O_t = (W_t, I_t, P_t)$. For instance, $(\mathrm{x\_}, -2, E)$ will represent an unstressed syllable set to an E that is approached by a descending leap of a third. In addition, $(W_t, -, G)$ will be used to represent a syllable with stress W_t set to the opening G, for which there is no preceding interval information.

It should be noted that the pitch representation chosen here is redundant, since all the pitch-related information is implicit in the pitch variable P_t alone, from which the interval variable I_t can be deduced. As it happens, however, this redundancy is helpful to HMM inference, since the framework has no other way of knowing, e.g., that the succession of symbolic values E–F and F–G represent the same diatonic interval. Nevertheless, our redundant representation creates an additional task for the model selection framework, namely to learn the appropriate pitch-interval constraints.

In Section 3.2, we already saw that the determination of HMM topology is one of the most difficult aspects of HMM inference. For the example of this article, we will be using *state splitting* to search the space of all candidate topologies. Here we will simply outline the search process; details will be presented elsewhere [8].

Table 1. Calculation of description lengths (DLs) for all the models considered in the HMM inference example of this section. The first column shows the candidate HMM's number of states n_S. The second column shows the state split from which that model was obtained. The next two columns contain DL values for the data and model respectively, and the fifth column lists the total DL. The last column lists the truncation parameter d used to encode the values of the HMM's parameters. The model selected by the MDL principle is the 10-state HMM, marked with an asterisk in Column 1.

n_S	State Split	*Data DL*	*Model DL*	*Total DL*	d
1	–	7128.60	152.886	7281.49	318
2	s_0	5228.79	368.049	5596.84	83
3	s_0	4213.64	661.950	4875.59	66
4	s_1	3372.82	634.118	4006.93	26
5	s_0	2879.03	667.795	3546.83	26
6	s_1	2381.20	754.486	3135.69	26
7	s_2	1892.71	844.757	2737.47	26
8	s_3	1578.80	981.823	2560.62	23
9	s_2	1336.01	1073.250	2409.26	23
*10	s_7	1278.16	1098.720	2376.88	22
11	s_7	1277.71	1193.230	2470.94	22
12	s_7	1277.71	1289.970	2567.68	22
13	s_7	1277.71	1388.810	2666.52	22
14	s_7	1277.71	1489.630	2767.34	22

Our topology search begins with a one-state HMM. At each subsequent stage, we perform an appropriately chosen state split, which increments the size of the model by one state. This process leads to a series of increasingly larger candidate HMM topologies. To monitor the search's progress, we calculate at each step the total description length (DL) of the data plus candidate model. The calculation uses the two-part code of eq. (3), with data DL given by the standard Shannon-Fano expression (4), and model DL given by expression (6). The state split chosen at each stage is the one that yields the model with smallest *total* DL at the next stage.

The results of the above process, as applied to our cadential formula modeling problem, are shown in Table 1. As can be seen from that table, the model DL—which measures a HMM's complexity—shows the expected increase as the number n_S of HMM states increases. At the same time, this complexity improves the model's capacity to fit the data; this is reflected by a concomitant decrease in data DL. For the first several stages of the search, the net result is an overall decrease in total DL. This indicates that, during these stages, each model can be considered a necessary refinement over its predecessor. The total DL eventually reaches a minimum value. From that point on, the increase in model complexity overpowers any benefit derived from improved goodness-of-fit. This heralds the onset of over-fitting, and the process terminates. The model that minimizes the total DL is selected as the winner. In our case, this is a ten-state HMM. The corresponding row in Table 1 is marked with an asterisk.

It should be clear from the above that, in addition to selecting the best model *overall*, the MDL principle also determines which state to split at *each stage* in the process. In this way, MDL addresses the difficult problem of searching efficiently over the space of HMM topologies by providing a *best-first* heuristic. This often allows us to interpret the intermediate stages as representing the best model at a particular level of refinement. In this spirit, we will list some intermediate HMMs that led up to the best one, identifying those aspects of the formula's behavior that were captured by each model. A fuller discussion of these intermediate models appears in the Online Supplement, Section 4.

- The **1-state model** simply implements the overall frequency of occurrence for

each output symbol in the data set.

- The **2-state model** captures the privileged role of G as a reference pitch in the formula.
- The **3-state model** also captures the role of pitches F and A as upper and lower neighbors to the G.
- The **4-state model** implements the distinction between the initial and final G's in the formula.
- The **5-state model** recognizes the special role of the note E; each HMM state now has a well-defined pitch interpretation.
- The **6-state model** implements pitch-stress rule (R1) of Section 4.1.
- The **7-state model** has learned the formula's overall *contour* $< G, F, E, F, G >$.
- The **8-**, **9-**, and **10-state models** show increasingly refined understanding of the pitch-stress constraints; the **10-state model** is chosen as the best.

The MDL-selected model of Figure 4 captures refined versions of pitch-stress rules (R1)–(R4). To see this, let us examine each of these rules in turn. Rule (R1) is represented by the fact that every HMM path must begin with the state sequence $< s_0, s_5, s_1 >$. As a result, all generated realizations of the formula must begin with a G that is typically stressed. Moreover, "typically" is here quantified to be "82% of the time," as reflected in the output probability of 0.82 that is assigned to the stressed syllable. The general form of (R2) is implemented in state transitions (s_8, s_3) and (s_4, s_2); these transitions carry the last stressed syllable of the formula most of the time, as can be seen by comparing appropriate transition and output probabilities on the graph. Likewise, the general form of (R3) is implemented in the transitions connecting states s_1, s_6, and s_4, including loops from a state to itself. The lower-probability transition/output sequences that violate rules (R2) and (R3) provide qualified exceptions to the latter; moreover, these sequences quantify the extent of the violation through their associated probabilities. Finally, rule (R4) is implicitly distributed throughout the graph, and is represented by those outputs that carry unstressed syllables. For instance, the loop on state s_1 represents extra G's that may be inserted in order to accommodate unstressed syllables near the beginning of the formula.

To complete our modeling goal for this section, we will now show that the HMM of Figure 4 can be used as a generative text-setting model. In other words, we will show how the HMM generates the formulaic realization corresponding to a *given* text. We first note that, when traversing a HMM path, our model simultaneously generates both text and melody, corresponding to the ordered triplet of variables (W_t, I_t, P_t). However, if the pattern of word-stress is given, we can simply constrain our HMM traversal to only those paths whose word-stress outputs match the W_t values of the given stress pattern. The interval and pitch outputs (I_t, P_t) collected along the path will then represent the generated melody. For instance, given the stress pattern < xx, x_, x_, x_, xx, x_ >, one possible HMM path consistent with that pattern is $< s_5, s_1, s_6, s_6, s_4, s_2, s_9, s_3 >$. Note the choice of "no syllable change" ($W_t =$ __) on the transition (s_2, s_9). We can therefore use that HMM path to generate the melody $< G, F, F, E, F, G, G >$. The reader is invited to check the details of the derivation.

This completes our example application of MDL to the Hidden Markov Model framework. We will conclude this article by discussing some implications of the MDL approach for the modeling of musical structure.

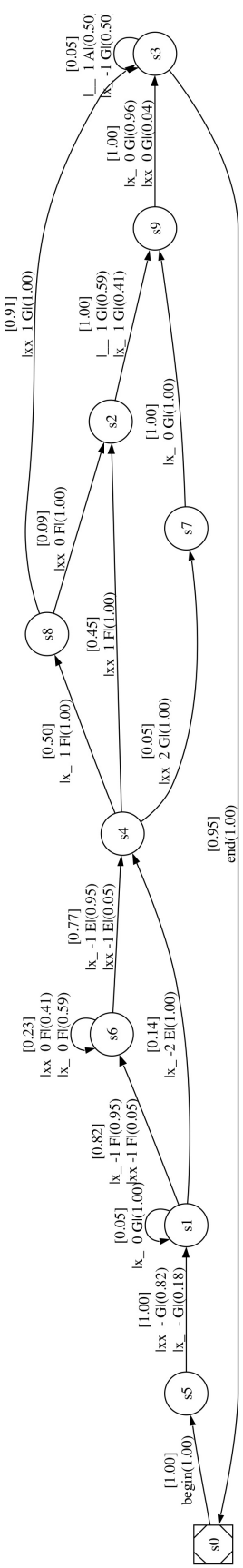


Figure 4. The ten-state HMM selected by the MDL principle in modeling the Greek chant cadential formula of Figure 3. Square brackets are used to mark transition probabilities on the arcs, while parentheses are used to mark output probabilities next to the corresponding output symbols.

5. The MDL Framework in Empirical Studies of Musical Structure

In this article, we outlined some key theoretical concepts and results in the Minimum Description Length framework, arguing that it provides a principled and precise model selection criterion for formal empirical studies of musical structure. We also saw the framework in action by applying it to the problem of HMM inference. In doing so, we hope to have offered a glimpse of the power and elegance of MDL as a general method of inductive inference from data. In this final section, we discuss the broader role that the MDL framework might play in empirical studies of music.

5.1. Musical Applications of MDL Beyond Hidden Markov Models

In recent years, MDL has gained increasing prominence in statistical and machine learning research. There are several reasons for this success. First, MDL effectively addresses over-fitting, offering theoretical justification in place of many previously employed ad hoc solutions. Second, MDL embodies and quantifies many of the model qualities that need to be negotiated in the process of model selection. As was discussed in [7], these qualities are not limited to goodness-of-fit and model simplicity, which are explicitly represented in the two-part code formulation. Third, MDL has broad scope of applicability in that it can be defined in principle for any family of models. Finally, MDL can often support efficient algorithmic implementations of model selection, as illustrated in this article by the state-splitting algorithm for HMM topology determination.

By focusing on how the framework specifically applies to HMM inference, we showed how a particular model family gives concrete form to MDL, as illustrated by the HMM complexity expression of eq. (6). We also showed how, by means of its MDL implementation, the HMM framework can be used to uncover syntactical relations between observation variables that provide formal representations of musical structure. In addition to the text-tune association explored in this article, the HMM framework is in fact capable of modeling a wide variety of syntactical phenomena in music; its expressive power as well as its limitations are explored in [8].

It should be clear, however, that the potential usefulness of MDL for music is not limited to Hidden Markov Modeling. The framework's powerful and elegant theoretical approach centers around general ideas of compact coding for data and parameter sets. Therefore, MDL is directly applicable to any parametrized family of probabilistic models. Moreover, the approach becomes particularly compelling when dealing with an unbounded parameter set and its associated propensity to over-fit the data. Other families of this type have been employed in the study of musical structure, one of the most notable examples being neural networks [29–31]. In these models, the size and complexity of the parameter set reflects the number and size of neural network layers, as well as their interconnections. The MDL principle could be used to determine both the parameter values and the connectivity of neural network models, in a manner analogous to that of HMMs.

Another model family that will most likely prove important for the study of musical structure is *probabilistic context-free grammars* [32, pp. 381–404]. Context-free grammars are also known as *Type II grammars* in Chomsky's hierarchy of formal languages [12, 13]. They are an extension of Finite State Machine grammars and, unlike the latter, can naturally express recursive hierarchical structure. In the context of music, it has been shown that these grammars can formalize a substantial portion of Schenkerian transformations [33]. The probabilistic version of

context-free grammars has already been explored in linguistics. In modeling musical structure, a probabilistic grammar of this type could offer several attractive possibilities, including: representing ambiguity in Schenkerian structure; exploring nuances of musical style; and identifying stylistic differences among composers. The inference of probabilistic context-free grammars from data is a much more complicated problem than the corresponding one for HMMs. However, some progress has been made in this area [34, 35]. Once again, the MDL framework can help identify the grammar's structure as well as its parameters.

When exploring a new model family through the MDL principle, one of the most direct implementations of the framework is the two-part code of eq. (3). The data description length term in this equation can be realized through the Shannon-Fano code expression (4), which applies universally to all probabilistic models. To complete the implementation, one would then need to develop an expression for the model description length term. This expression would reflect the structure of the model family under investigation and could be established in a manner analogous to the derivation of eq. (6).

Finally, given the rising prominence of Bayesian methods in recent computational studies of music structure and cognition [19, 36, 37], it is important to add that there exist deep connections between the MDL principle and Bayesian approaches to model selection. These connections will be developed in the Online Supplement, Section 5.2.

5.2. The Role of MDL in Building and Evaluating Theories of Music

We will conclude with an attempt to place MDL within the context of building and evaluating theories of music. First of all, it should be stressed that model selection—particularly in the narrow algorithmic sense advocated in this article—should not be equated with the broader process of music theoretic discovery. Music theorists have many different ways of formulating and solving problems. In particular, a music theorist may want to explore different rule formats, or may be able to formalize certain components of a theory but not others.

To understand why this flexibility is desirable, it is instructive to compare the generative text-setting model of Fig. 4 with rules (R1)–(R4) of Section 4.1. However approximate—and perhaps partly inaccurate—the latter rules may be, they seem to reflect a representation of knowledge that is directly accessible to people. In particular, such rules are easier to understand, communicate, and perhaps relate to work in other problem domains. In some essential way, they appear to promote our understanding of the problem. By contrast, even though a HMM may offer an accurate and nuanced generative model, its content is not always easy to understand directly. It is often fairly difficult to process and communicate the detailed knowledge contained in all its nodes, arcs, outputs, and their associated probabilities.

These observations are not meant to undermine the usefulness of HMMs and similar formal representations. Rather, they serve to clarify their possible role in building music theories. As generative grammars, HMMs serve their purpose rather well, in that they successfully circumscribe the corpus or behavior under investigation. Moreover, they do so in a precise and fine-grained way. As such, HMMs illustrate what can be typically accomplished by MDL-derived formal models. The knowledge that goes into building such models must be extracted from the corpus in a way that is not only efficient, but also unbiased. In out approach, this is achieved through algorithmic corpus analysis that extracts statistical regularities, and that is assessed quantitatively at each stage through MDL. This initial com-

puter processing of low-level information performs a task that may be too tedious, and also unreliable, for a human to do. The result is a formal model that is not only precise, but also helps "visualize" significant patterns that are not always easy to observe directly in the corpus. Therefore, this model forms a valuable starting point for a next level of analysis. In this next level, the theorist *interprets* the model's structure, in order to obtain more human-oriented formulations of the knowledge that it embodies.

As an example of this process of interpretation, consider Mavromatis's study of Greek church chant [6]. A HMM analysis along the lines presented in this article led to a set of very complicated models, one for each melodic mode of the idiom. Mavromatis proceeded to interpret each model by analyzing the melodies it generates on a *given* text. This analysis revealed how Greek chant melody can be broken down into formulaic segments, each of which can be used in many different contexts. These segments were not always directly visible in the complex HMM graphs. Rather, they were extracted as a human-level representation of the chant's structure, which can be analyzed and tabulated very much like motivic content, and which could even guide a pedagogical approach to chant improvisation.

There are, of course, many different ways in which researchers could exploit formal models like HMMs, while pursuing empirical theories of musical structure. A minimal requirement of the empirical approach is that the resulting theories are formulated precisely enough to be falsifiable. Beyond that, the music theoretic process of discovery is creative, imaginative, and open-ended. Just like scientific discovery in general, it is unlikely that it can be captured by a simple algorithmic process [38].

To summarize, MDL modeling of musical structure can provide an important ingredient in the empirical study of music. Through the principle of compact coding, MDL guides the extraction of essential patterns from a corpus of musical data. The resulting model offers a compact and computable representation of the knowledge contained in these patterns, which researchers can then analyze and interpret using a wide variety of formalisms and approaches. Since MDL embodies important model selection values, theories built in this way can more readily address the scientific community's broad epistemological concerns. We believe that the MDL principle offers a widely applicable, flexible, and powerful framework in which empirical studies of music can be formulated and engaged in fruitful new ways.

References

[1] N. Cook, *"Computational and Comparative Musicology"*, in *Empirical Musicology: Aims, Methods, Prospects*, E. Clarke and N. Cook, eds., Oxford University Press, Oxford; New York, 2004, chap. 6, pp. 103–126.

[2] M.G. Brown *Explaining Tonality: Schenkerian Theory and Beyond*, University of Rochester Press, Rochester, NY, 2005.

[3] J. Rissanen *Stochastic Complexity in Statistical Inquiry*, Series in Computer Science Vol. 15, World Scientific, Singapore; New Jersey; London, 1989.

[4] P.D. Grünwald *The Minimum Description Length Principle*, Adaptive Computation and Machine Learning MIT Press, Cambridge, MA, 2007.

[5] C.M. Bishop *Pattern Recognition and Machine Learning*, Springer Series on Information Science and Statistics Springer, New York, 2006.

[6] P. Mavromatis, *"A Hidden Markov Model of Melody Production in Greek Church Chant"*, Computing in Musicology 14 (2005), pp. 93–112.

[7] ———, *The* Echoi *of Modern Greek Church Chant in Written and Oral Transmission: A Computational Model and its Cognitive Implications.*, Ph.D. Dissertation, Eastman School of Music, University of Rochester, 2005.

[8] ———, *"Building Probabilistic Musical Grammars Using Hidden Markov Models"*, (forthcoming).

[9] L.R. Rabiner and B.H. Juang, *"An Introduction to Hidden Markov Models"*, IEEE ASSP Magazine 3 (1986), pp. 4–16.
[10] L.R. Rabiner, *"A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition"*, Proceedings of the IEEE 77 (1989), pp. 257–286.
[11] O. Cappé, E. Moulines, and T. Rydén *Inference in Hidden Markov Models*, Springer Series in Statistics Springer, New York, 2005.
[12] J.E. Hopcroft and J.D. Ullman *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.
[13] N. Chomsky, *"On Certain Formal Properties of Grammars"*, Information and Control 2 (1959), pp. 137–67.
[14] R. Wilhelm and D. Maurer *Compiler Design.*, Addison-Wesley, Reading, MA, 1995 Translated by Stephen S. Wilson.
[15] J.P. Bello and J. Pickens, *"A Robust Mid-level Representation for Harmonic Content in Music Signals"*, , in *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR-05)* , London, UK, 2005.
[16] D. Temperley, *"A Bayesian Approach to Key Finding"*, in *Music and Artificial Intelligence: Proceedings of the Second International Conference, ICMAI 2002, Edinburgh, Scotland*, C. Anagnostopoulou, M. Ferrand and A. Smaill, eds., Springer, Berlin, 2002, pp. 195–206.
[17] C. Raphael and J. Stoddard, *"Functional Harmonic Analysis Using Probabilistic Models"*, Computer Music Journal 28 (2004), pp. 45–52.
[18] D. Temperley, *"Bayesian Models of Musical Structure and Cognition"*, Musicae Scientiae 8 (2004), pp. 175–205.
[19] D. Temperley *Music and Probability*, MIT Press, Cambridge, MA, 2007.
[20] A. Stolcke and S.M. Omohundro, *"Hidden Markov Model Induction by Bayesian Model Merging"*, in *Advances in Neural Information Processing Systems 5*, S.J. Hanson, J.D. Cowan and C.L. Giles, eds., Morgan Kaufmann, San Mateo, CA, 1993, pp. 11–18.
[21] ———, *Best-first Model Merging for Hidden Markov Model Induction*, TR-94-003, International Computer Science Institute, Berkeley, CA, 1994 Available at *http://www.icsi.berkeley.edu/ftp/global/pub/techreports/1994/tr-94-003.pdf* (last visited August 2008).
[22] M. Ostendorf and H. Singer, *"HMM Topology Design Using Maximum Likelihood Successive State Splitting"*, Computer Speech and Language 11 (1997), pp. 17–41.
[23] M.Y. Chen, *"Toward a Grammar of Singing: Tune-Text Association in Gregorian Chant"*, Music Perception 1 (1983), pp. 84–122.
[24] J. Halle and F. Lerdahl, *"A Generative Textsetting Model"*, Current Musicology 55 (1993), pp. 3–23.
[25] B. Hayes and A. Kaun, *"The Role of Phonological Phrasing in Sung and Chanted Verse"*, The Linguistic Review 13 (1996), pp. 243–303.
[26] B. Hayes, *"Textsetting as Constraint Conflict"*, in *Towards a Typology of Poetic Forms*, J.L. Aroui and A. Arleo, eds., Elsevier Science, Amsterdam, 2008.
[27] M. Liberman, *The Intonational System of English*, PhD Dissertation, MIT, Cambridge, MA, 1975.
[28] A.L. Vallindras (ed.) *Anastasimatarion*, Ekdoseis Zoi, Athens, Greece, 1998.
[29] P.M. Todd and D.G. Loy (eds.) *Music and Connectionism*, MIT Press, Cambridge, MA, 1991.
[30] P. Toiviainen, *Modelling Musical Cognition with Artificial Neural Networks*, PhD Dissertation, University of Jyväskylä, Jyväskylä, Finland, 1995.
[31] N. Griffith and P.M. Todd (eds.) *Musical Networks: Parallel Distributed Perception and Performance*, MIT Press, Cambridge, MA, 1999.
[32] C.D. Manning and H. Schütze *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, MA, 1999.
[33] P. Mavromatis and M. Brown, *"Parsing Context-Free Grammars for Music: A Computational Model of Schenkerian Analysis"*, in *Proceedings of the 8th International Conference on Music Perception and Cognition, Evanston, IL*, S.D. Lipscomb, R. Ashley, R.O. Gjerdingen and P. Webster, eds., Causal Productions, Adelaide, Australia, 1994.
[34] A. Stolcke, *Bayesian Learning of Probabilistic Language Models*, PhD Dissertation, University of California, Berkeley, CA, 1994.
[35] P.D. Grünwald, *"A Minimal Description Length Approach to Grammar Inference"*, in *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, S. Wermter, E. Riloff and G. Scheler, eds., Springer-Verlag, Berlin, Heidelberg, New York, 1996, pp. 203–216.
[36] A.T. Cemgil, P. Desain, and B. Kappen, *"Rhythm Quantization for Transcription"*, Computer Music Journal 24 (2000), pp. 60–76.
[37] C. Raphael, *"A Hybrid Graphical Model for Rhythmic Parsing"*, Artificial Intelligence 137 (2002), pp. 217–38.
[38] P. Langley, H.A. Simon, G.L. Bradshaw, and J.M. Zytkow *Scientific Discovery: Computational Explorations of the Creative Processes*, MIT Press, Cambridge, MA, 1987.