ABSTRACT

This paper presents a probabilistic formulation of music language modelling based on the generative theory of tonal music (GTTM) named probabilistic GTTM (PGTTM). GTTM is a well-known music theory that describes the tree structure of written music in analogy with the phrase structure grammar of natural language. To develop a computational music language model incorporating GTTM and a machine-learning framework for data-driven music grammar induction, we construct a generative model of monophonic music based on probabilistic context-free grammar, in which the time-span tree proposed in GTTM corresponds to the parse tree. Applying the techniques of natural language processing, we also derive supervised and unsupervised learning algorithms based on the maximal-likelihood estimation, and a Bayesian inference algorithm based on the Gibbs sampling. Despite the conceptual simplicity of the model, we found that the model automatically acquires music grammar from data and reproduces time-span trees of written music as accurately as an analyser that required elaborate manual parameter tuning.

Index Terms— Statistical music language model, GTTM, PCFG, time-span tree analysis, statistical grammar induction

1. INTRODUCTION

Music transcription is a challenging topic in music processing. For this, acoustic modeling of musical sounds has been widely studied [1]. However, this is not sufficient to remove all ambiguities of transcription due to acoustic variations, and using prior knowledge on the output musical score is in order. The same situation happens in speech recognition, where the knowledge on linguistics is incorporated in the language model. The purpose of this study is to construct a model that can capture the grammar of written music.

Musical pieces have a hierarchical structure. Notes are grouped into motives, which are grouped into phrases, which are grouped into musical sentences (or passages), which are grouped into sections, etc. In addition, some notes are more salient than adjacent notes. This structure of music resembles the phrase structure of natural language [2], which can be represented as a syntax tree. The generative theory of tonal music (GTTM) [3] attempts to model such a hierarchical structure of music. GTTM proposes to use a tree representation called the time-span tree to describe the relative importance of notes. The time-span tree can be interpreted as a simplification or reduction process of a musical passage (Fig. 1). GTTM also proposes rules to derive a reasonable time-span tree for a given musical passage, which combine musical knowledge such as counterpoint theory [4], harmonic theory [5], and Schenkerian analysis [6].

Fig. 1. Example of time-span tree and the reduction and generation of a musical passage (top). The two types of production rules in the probabilistic GTTM (bottom).

The computational formulation of GTTM has been studied in the last decades [7–10] because the prescribed rules of GTTM are not quantitatively or computationally formalised. Based on parameterisation of the rules in GTTM, an automatic time-span tree analyser (ATTA) was implemented [7]. The adjustment of the 46 parameters remained as a problem. Recently a probabilistic method for time-span tree analysis called $\sigma$GTTM III is proposed, which can learn some parameters from labelled music data [10]. There are other notable studies on automatic music analysers [11, 12]. Although these studies provide grouping structure analysis, metrical analysis, and reduction analysis, they are not a generative model and cannot be directly applied to music transcription [1] or automatic music generation and arrangement [13]. Moreover, these studies have not taken advantage of the developed machine-learning techniques in natural language processing (NLP) [14–17].

To solve these problems, we formulate a probabilistic generative model of music based on GTTM named the probabilistic GTTM (PGTTM). We interpret that the time-span tree represents a production process of musical notes and formulate PGTTM with probabilistic context-free grammar (PCFG) [18, 19]. In contrast to the phrase structure grammar of linguistics where an intermediate node corresponds to grammatical categories such as ‘NP’ and ‘VP’, a node of the time-span tree represents a musical note (Fig. 1). In addition, two types of production rules must be distinguished to indicate the relative importance of children notes, and there are constraints on the pitches and note values (lengths) to consistently describe the reduction/generation process.

There are other works that applied PCFG models for music. The idea of using PCFG for the probabilistic formulation of GTTM is already explored in $\sigma$GTTM III [10], which is not fully formulated.
as a generative model of musical notes and requires other analysing
tools [7, 8] to obtain the time-span trees. A two-dimensional
structure model was proposed [20] and applied for multi-pitch anal-
ysis. Our study shares the same interests in unsupervised learning of
music language model. A PCFG model for harmonic analysis was
proposed in [21]. The main contribution of this study is to provide
an explicit construction of a statistical music language model that
integrates GTTM and a machine-learning framework for inducing
music grammar from data. In this study, we confine ourselves to
monophonic music.

We evaluate the proposed PGTTM in terms of the accuracy of
time-span tree analysis. We examine the performance of the model
with both supervised and unsupervised learning, and compare the re-
sults with previously proposed analysers. We found that the PGTTM
performs as accurately as ATTA. This is encouraging as PGTTM is
contemporarily simple and does not require elaborate parameter tun-
ing. Error analyses show that whereas local tree structures were rel-
avely well estimated with the PGTTM, the higher-level structures
were poorly estimated. The result suggests the importance of in-
troducing latent grammatical categories into the PGTTM. Ideas for
further refinements are given at the end.

2. PROBABILISTIC GTTM (PGTTM)

This section explains the proposed model.

2.1. Basic model

Each musical note is represented as a pair \((p, r)\) of pitch \(p\) and note
value \(r\). We can use either spelled pitches or integral pitches, and
the set of possible pitches is denoted by \(\Omega_p\). A rest is represented
by a symbol ‘\(\cdot\)’, which we also include in \(\Omega_p\). We define the note
value as the score-written length of the note relative to a whole note
(a quarter note has a value of \(1/4\), a dotted half note has a value of
\(3/4\), etc.). The set of possible note values is denoted by \(\Omega_r\). Then
a monophonic passage can be represented as a sequence \((p_n, r_n)\)\(^{1=1}\),
where \(N\) is the number of notes and rests.

A PCFG model [18] is represented with the set of terminals \(\Omega_T\),
the set of non-terminals \(\Omega_S\), which contains the start symbol \(S\), and
the set of production rules \(R\). In the Chomsky normal form, a pro-
duction rule \(A \rightarrow a\) (\(A \in \Omega_S\), \(a \in \Omega_N \times \Omega_N \cup \Omega_T\)) is associated
with a probability value \(P(A \rightarrow a)\). The symbol \(A\) and the symbols
in \(a\) in a production rule will be called the parent and the children,
respectively. Given a sequence of non-terminals \(w = w_1 \cdots w_N\)
\((w_n \in \Omega_S\), a set of production rules to derive \(w\) from \(S\) is called a
derivation of \(w\), which can be represented as a parse tree (Fig. 1).

In the probabilistic formulation of GTTM, we need the follow-
ing modifications and extensions of this ordinary PCFG. First, the
parent and the children of a production rule should be represented
as musical notes. An exception is the most top production of notes
from the start symbol, which will be explained in the next paragraph.
Second, to describe the structure of the principal note vs. the sub-
dordinate note, we distinguish two types of production rules; one has
the principal note on the left and the other has it on the right. Then a
production rule can be written in the form \((p, r) \rightarrow s(p_l, r_l)(p_r, r_r)\)
where \(p, p_l, p_r \in \Omega_p\), \(r, r_l, r_r \in \Omega_r\), and \(s = L, R\) indicates that
\((p, r)\) is the principal note. Finally, since the time span must be
conserved in producing a note, we must have \(r = r_l + r_r\), and thus
we can write \(r = r - r_l\). The fact that the principal note has the
same pitch as the parent note requires \(p = p_s\) (Fig. 1).

Let us now define the PGTTM. The production process begins with
the start symbol \(S\) with a note value \(r_s\), which corresponds to
the total note value of a musical passage. The production rules are
either of the form \((S, r_s) \rightarrow s(p_l, r_l)(p_r, r_r - r_l)\) or \((p, r) \rightarrow
s(p_l, r_l)(p_r, r - r_l)\) where \(p, p_l, p_r \in \Omega_p\), \(r, r_l, r_r \in \Omega_r\), and
\(s = L, R\). The associated probabilities are written as
\(P((S, r_s) \rightarrow s(p_l, r_l)(p_r, r_r - r_l))\) and \(P((p, r) \rightarrow s(p_l, r_l)(p_r, r - r_l))\),
which satisfy the following normalisation conditions:

\[
\sum_{s \in \Omega_p, p \in \Omega_p, r \in \Omega_r} P((S, r_s) \rightarrow s(p_l, r_l)(p_r, r_r - r_l)) = 1, \quad (1)
\]

\[
\sum_{s \in \Omega_p, p \in \Omega_p, r \in \Omega_r} P((p, r) \rightarrow s(p_l, r_l)(p_r, r - r_l)) = 1. \quad (2)
\]

There is a constraint that \(P((p, r) \rightarrow s(p_l, r_l)(p_r, r - r_l)) = 0\)
unless \(p = p_s\) as explained above, and we also assume that rests
must not be a principal note so that the probability is zero unless
\(p \in \Omega_p \setminus \{R\}\). A set of production rules that yields a given musical
passage is called a time-span tree.

The production probabilities strongly depend on the key of the
musical passage, and this dependence can be theoretically described
by first considering a different PGTTM for each of 24 keys (major
keys and minor keys on 12 pitch classes) and then constructing their
mixture model. In the following we consider a situation that the key
is known in advance, so all passages can be transposed to C major or
C minor.

2.2. Simplifications and refinements

Simplifications of the PGTTM described in Section 2.1 are necessary
if we are to derive computationally tractable inference algorithms.
First, we assume that the production rule probabilities are indepen-
dent for pitch and note values. This means that the production rule
probability is factorised as follows:

\[
P((p, r) \rightarrow s(p_l, r_l)(p_r, r_r - r_l)) = P^s(p)P^p(p \rightarrow s(p_l, r_l)(p_r, r_r - r_l)) \quad (3)
\]

where \(\sum_{p} P^s(p) = 1\), \(\sum_{p \in \Omega_p, p \in \Omega_p} P^p(p \rightarrow s(p_l, r_l)(p_r, r_r - r_l)) = 1\), and
similarly for the production rules from \(S\). Second, we represent the pitch with an
integral pitch class, and thus \(\Omega_p = \{C, C\#, \cdot \cdot \cdot, B, R\}\) (where
\(C\# = D\) etc.). We will use the following notations: \(\phi_s = P^s(p)\),
\(\Theta_{\Omega_p, p, r} = P^p(p \rightarrow s(p_l, r_l)(p_r, r_r - r_l))\), and \(\Theta = \{\phi_s, \Theta_{\Omega_p, p, r}\}\).
Finally, we assume that the probabilities for note values are scale
invariant so that \(P^*(r \rightarrow s(p_l, r_l)(p_r, r_r - r_l))\) is a function of \(r_l/r\) and not especially
dependent of \(r_l\) and \(r_r\). The probability of a time-span tree \(T\) is given as

\[
P(T|\Theta) = \prod_{s \in \Omega_p, p \in \Omega_p, r \in \Omega_r} \phi_s \Theta_{\Omega_p, p, r}^\phi c((p, r) \rightarrow s(p_l, r_l)(p_r, r_r - r_l)|T) \quad (4)
\]

where \(c((p, r) \rightarrow s(p_l, r_l)(p_r, r_r - r_l)|T)\) is the number of times that the
production rule \((p, r) \rightarrow s(p_l, r_l)(p_r, r_r - r_l)\) appears in \(T\).

It has been noted that the choice of the principal note is influ-
enced by the magnitude relation of metrical weights of the relevant
note pair (Rule TSRPR 1 in [3]). Here, the metrical weight of a note
indicates the relative strength of the metrical (beat) position of its
onset [22]. To incorporate this preference rule, we allow that the
probability \(P^s(p)\) depends on the relative metrical weights of the
relevant note pair. If we represent the metrical weights of note \((p_l, r_l)\)
and \((p_r, r_r)\) as \(\omega_L\) and \(\omega_R\), \(P^s(p)\) take different values depending on
whether \(\omega_L/\omega_R\) is less than, equal to, or greater than unity.

2.3. Bayesian formulation of the PGTTM

As we explain in Sec. 3, we can derive an unsupervised learning
algorithm for PGTTM based on the maximal-likelihood principle.
An alternative way of unsupervised learning is Bayesian inference
[14]. With the Bayesian formulation, we can develop a Monte-Carlo
3. INFERENCE ALGORITHMS FOR PGTTM

Basic inference algorithms for PCFG have been well-developed. With some modifications, we can derive inference algorithms for the PGTTM. This section summarises the results.

3.1. CYK-Viterbi, inside, and outside algorithms

Given a musical passage $W = (p_o, r_o)_{n=1}^N$, the most probable time-span tree can be obtained with the CYK-Viterbi algorithm, which is a dynamic programming. We will use the notation $W_n = (p_o, r_o), W_n^m = W_n \cdots W_m, \text{and } r_n^m = r_n + \cdots + r_m$. We introduce a variable $\gamma_{n,m,p} = \max P(T)$ for $1 \leq n \leq m \leq N$ and $p \in \Omega_p$ where $T$ denotes a time-span tree with parent $(p, r_n^m)$ and yield $W_n^m$ (we express this as $(p, r_n^m) \xrightarrow{T} W_n^m$). This can be calculated recursively as

$$\gamma_{n(n+1)p} = \max_{k,s,p,r,R} \left\{ P((p, r_n^k) \rightarrow s(p_L, r_n^k)(p, r_n^k)) \right\}\left(\gamma_{n(k+1)p}, \gamma_{k(n+1)p}\right)$$

for all $p \in \Omega_p$ with the initial values $\gamma_{n,m,p} = \delta_{W_n} (\delta$ denotes Kroenecker’s delta). Then the probability of the most probable time-span tree is given as $\gamma_{1,N_S}$. In the calculation of Eq. (4), the arguments that yield the maximal probability are also memorised as $\delta_{W_n}, \delta_{W_n}, \text{etc.}$ Once all $\gamma_{n,m,p}$ have been computed, we can obtain the most probable time-span tree by expanding the tree according to these arguments starting from $(S, r_1^N)$, as in the back-tracking procedure of the Viterbi algorithm for hidden Markov model.

To learn parameters of the PGTTM, we calculate the inside variables $\beta_{n,m,p}(W)$ and outside variables $\alpha_{n,m,p}(W)$ defined as

$$\beta_{n,m,p}(W) = \sum_{T,(p, r_n^m) \xrightarrow{T} W_n^m} P(T)$$

$$\alpha_{n,m,p}(W) = \sum_{T,(S, r_1^N) \xrightarrow{T} W_1^M} P(T)$$

(5) and (6) respectively.

If the dependency on $W$ is self-evident, we simplify the notation as $\beta_{n,m,p}$ and $\alpha_{n,m,p}$. These quantities can also be computed by dynamic programming algorithms called the inside and outside algorithms. For the PGTTM, the inside algorithm has the following update equation similar to Eq. (4) except with the maximisation $\max_{k,s,p,r,R}$ replaced by a summation $\sum_{k,s,p,r,R}$. The outside algorithm is based on the following recursive equation:

$$\alpha_{n,m,p}(W) = \sum_{s,q,p_R} P(q, r_n^m \rightarrow s(p_L, r_n^m)(p, r_n^m)) \alpha_{nq} \beta_{(m+1)kp_R}$$

+ $\sum_{s,q,p_R} P(q, r_n^m \rightarrow s(p, r_n^m)) \alpha_{nq} \beta_{(n+1)kR}$

(7)

with initial values $\alpha_{1,N_S} = 1$ and $\alpha_{N_p} = 0 (p \neq S)$. In practice, the left-hand side of Eq. (7) can be computed more efficiently by applying the constraints explained below Eq. (2).

3.2. EM algorithm for maximal-likelihood estimation

Given a set of (unlabelled) data $W$, the probability parameters can be estimated by the maximal-likelihood principle: $\Theta = \arg \max_\Theta P(W|\Theta)$. The expectation-maximisation (EM) algorithm can be derived by the following iterative minimisation [23]:

$$\Theta_{new} = \arg \max_\Theta \sum_T P(T|W, \Theta^{old}) \log P(T, W|\Theta).$$

By differentiating $Q(\Theta, \Theta^{old})$ with respect to $\phi_s, \theta_{spL,P_R}$, and $\rho_{SR}$, the following updating equations are derived:

$$\phi_{new} = \frac{1}{C_0} \sum_{p_L, p_R, r_L} Q((p, r) \rightarrow s(p_L, p_R, r_L); \Theta^{old})$$

$$\theta_{new} = \frac{1}{C_0} \sum_{p_L, p_R, r_L} Q((p, r) \rightarrow s(p_L, p_R, r_L); \Theta^{old})$$

$$\rho_{new} = \frac{1}{C_0} \sum_{p_L, p_R, r_L} Q((p, r) \rightarrow s(p_L, p_R, r_L); \Theta^{old})$$

(8)

(9)

(TW denotes the set of all possible time-span trees of W.) The quantity in Eq. (9) is given as

$$\phi_{old} \theta_{spL,P_R} \rho_{SR} \sum_{n=1}^N \sum_{m=1}^n \sum_{k=1}^{m-1} \alpha_{old}(W) \beta_{old}(W) \gamma_{old}(W)$$

$$\alpha_{old}(W) \theta_{old}(W) \rho_{old}(W)$$

(10)

where $\beta_{old}$ is the inside (outside) variable calculated with $\Theta^{old}$. The case for production rules with parent $(S, r_S)$ is similar.

3.3. Bayesian inference algorithm using Gibbs sampling

Given data $W$ and hyperparameters $\Lambda = (\eta, \lambda_s, \lambda_R, \nu_{sp}),$ the goal of the Bayesian learning is to estimate the posterior distribution of the model parameters $P(\Theta|W, \Lambda)$, which generally cannot be solved analytically. Fortunately, we can develop a Monte-Carlo method to draw samples from $P(\Theta|W, \Lambda)$ (T is a time-span tree) via the Gibbs sampling [14].

The Gibbs sampling method is based on alternating samplings of the probabilities $P(\Theta|T, W, \Lambda)$ and $P(T|W, \Lambda)$. The former probability can be written as a product of Dirichlet distributions as $P(\Theta|T, W, \Lambda) = P_{Diag}(\Theta)$ where

$$f_S(T) = \sum_{p_L, p_R, r_L} c((p, r) \rightarrow s(p_L, p_R, r_L); T)$$

(11)

and similarly for $\lambda_s$ and $\nu_{sp}$. Therefore $\Theta$ can be sampled from $P(\Theta|T, W, \Lambda)$ by sampling from the Dirichlet distributions.

Next, given a set of sampled parameters $\Theta$, a time-span tree $T$ can be sampled from $P(T|\Theta, W, \Lambda) = P(T|\Theta, W) \propto \sum_{W} P(W)$ by the following recursive sampling for each node. Each node is indicated as a triplet $(n, m, p)$ $1 \leq n \leq m \leq N$, $p \in \Omega_p$ meaning that the symbol $p$ spans the time span $r_n^m$. Starting from the top node $(1, N, S)$, if there is a node $(n, m, p)$ with $n < m$, it is expanded by sampling $s, k, p_L, p_R$ from the distribution $P((p, r_n^m) \rightarrow
4. EVALUATION

4.1. Evaluation setup

To test PGTTM, we implemented a time-span tree analyzer based on the CYK-Viterbi algorithm and evaluated the analyzer with a database of manually labeled time-span trees. The database consisted of 300 musical passages with time-span trees analyzed by a music expert [9].

To test its ability to learn the parameters, we evaluated the PGTTM in different learning conditions. The first two conditions were supervised learning, one the open condition (piece-wise cross validation) in which the training data did not include the test data and the other the closed condition in which the test data was also used for learning. To avoid zero frequencies, we uniformly added a 0.1 count for all frequencies. The next conditions were unsupervised learning, one based on the EM algorithm (Sec. 3.2) and the other based on the Gibbs sampling method (Sec. 3.3). For the EM algorithm, the initial parameter values were randomly chosen. For the Gibbs sampling method, all hyperparameters were set as 0.1 and after sampling, the parameter set that yielded the maximal likelihood was chosen and applied EM iterations before it was used for the time-span tree analysis.

An estimated time-span tree was compared with the reference in the database and the accuracy was calculated in the following way. First a node of an estimated time-span tree was defined as matched if it had a corresponding node with the same parent and the same children in the reference tree. Then the accuracy was defined as the ratio of the number of matched nodes to the total number of nodes.

4.2. Evaluation results and error analysis

The results in Table 1 show that the accuracies for the PGTTM were nearly equal to that for ATTA. For reference, the accuracy is also shown for \( \sigma \)GTTM III, which uses the true group boundary of notes in the annotated data and thus cannot be fairly compared to the other algorithms. The fact that the open and closed tests for supervised learning had nearly equal accuracies indicates that there was little data sparseness and the accuracy would not improve much with a larger data size. The EM algorithm and the Gibbs sampling had similar accuracies that are lower than the case of supervised learning.

\[
s(p_L, r_n^k, r_n^p | p_R, r_n^m, W, \Theta) = \frac{\beta_{kmp} \beta_s r_n^k p_{r_n^m} | s(r_n^m) (r_n^k)^k} {\sum_{m} \beta_{kmp} \beta_s r_n^k p_{r_n^m} | s(r_n^m) (r_n^k)^k}
\]

4.3. Discussion

It is encouraging that the PGTTM worked as accurately as ATTA without elaborate manual tuning of parameters. However the results also suggest refinements of the model are necessary to improve the accuracy. First, as in the case with NLP, \( \sigma \)创作 rule probabilities depend rather strongly on the context, and it would be important to incorporate dependence between several notes in successions into the model [16]. Another important issue is the choice of symbols in the grammar. Since we have not essentially used non-terminals, the production rules of PGTTM are the same for all positions and heights in the time-span tree. The fact that notes with higher metrical weight are often more important in the lower-height nodes but the clauses are often on weak beats suggests that we need to extend the model with latent symbols, which would improve the time-span tree analysis for nodes with higher heights. Symbol refinement used for NLP [19, 24] can be an aid for this extension.

5. CONCLUSION

Based on GTTM, we have constructed a probabilistic tree structure model of written music. We formulated the PGTTM based on an extension of PCFG, for which both supervised and unsupervised learning techniques can be applied. Despite the conceptually simple construction of the model, the PGTTM produced musical syntactic parsing as accurately as the previously proposed method with elaborate manual tuning of parameters. The results suggest further refinements of the grammatical model. For future work, we plan to apply the model for music transcription and automatic music arrangement and extend the model for polyphony.
6. REFERENCES


