社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－MUS－77 (6)
2008／9／22

構造を持つ楽曲データを対象とした質問学習にもとづく楽曲生成

木村 誠 †， 土井 晃一郎 †， 山本 章博 †

† 京都大学大学院情報学研究科 知能情報学専攻

あらまし 本研究では，機械によって楽曲を生成し，人間の行う作曲行為を支援するという目標のため，人間の意図する楽曲構造の機械学習を行う．人間が他の誰かに作曲をさせる場合，作曲させる側は自分のイメージを相手に伝え，曲を作る側は伝えられた情報に基づいて作曲を行い，作られた曲に関して作曲をさせる側がまた意見を言い，とさまざまな情報のやり取りの末に曲が作られる．本研究ではこれを質問学習モデルでモデル化し，実際に楽曲の生成を行いながら学習を進める質問学習システムの設計を行った．また，設計したシステムの実装と試用を行い，評価を得た．

Music Generation based on Query Learning for Structured Music Data

Makoto KIMURA † Koichiro DOI † Akihiro YAMAMOTO †

† Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University

Abstract In this research we propose machine learning of music structure in order to develop a computer program helping human beings compose music. When a person asks another to compose a music, he would tell his idea on the music and the composer would design a music with refining the idea and returning queries to the requesting person. This means that a music is completed after some interaction between the two persons. In the present paper, we model the task in the framework of query learning and design a learning system which generates music.Then we implement a learning system and evaluate the system with impression of people try to use it.

1 はじめに

近年はインターネットの発達により，Webサイトやブログでの情報発信・動画共有サイトへの動画投稿など，私たちが個人でコンテンツを作成し発信する機会が増加している．Webサイトや動画といったコンテンツの中ではBGMとして楽曲が用いられることがあり，また，楽曲そのものがコンテンツとなることもある．しかし楽曲は著作権で保護されており，みだりに他人の作った楽曲を使用することはできない．そこで，自らに著作権があるオリジナルな楽曲や，パブリックドメインの誰でも利用できる楽曲が必要とされ，そのため作曲という行為をより手軽に行うためのシステムが望まれる．

本来，作曲は専門的な知識が要求されるものであり，また労力もかかる．自動作曲や自動編曲ついては古くから多くの研究がなされてきた [1] [4] が，音楽の形式的な評価は難しく，最終的には生成される曲を人間が評価することによってに作曲(編曲)システムを評価せねばならず非常に扱いにくい問題である．

本研究では，この作曲という行為を質問学習モデル [2] [7] の枠組みでモデル化し，作曲をさせる側とする側での情報のやり取りとして捉える．質問学習モデルとは計算論的学習において提唱されているモデルの一つであり，学習を行う際に，ユーザから与えられる情報だけを用いるのではなく，機械から積極的に対象について質問を許す，というモデルである．さらに多くの場合，仮説を抽象的な対象から具体的なものへあるいはその逆というように，具体化・一般化によって学習を進める．本研究でも作曲という行為を，抽象的なイメージを具体化していき，より精緻なデータにしていくことであると捉える．そして，作曲をさせる側をユーザ，実際に楽曲を作る側を機械としてユーザと機械の間でさまざまな情報をやり取りしながら，学習を進める質問学習システムの設計を行った．

なお本稿は研究の初期段階であり，生成する楽曲は単音の旋律のみからなるものに限定する．しかし，質問学習モデルで作曲をするに当たって一番本質的な部分を含んでいると考えている．

本稿の構成を以下に示す．第2章では準備とし

て，質問学習に関する説明と，対象とする音楽データに関する説明を行う．第3章では設計した学習アルゴリズムについての説明を行う．第4章ではアルゴリズムを試用してもらった結果について述べ，最後に第5章でまとめと考察を行う．

2 準備

2.1 質問学習モデル

機械学習問題を定義するには，以下の6つの要素が必要であるとされている [6]．

1. 学習対象: 学習される対象がどのような集合であるかを指す．本研究では構造を持つ楽曲データの集合がこれに当たる．
2. 学習プロトコル: 学習を行う際，教師と生徒(ユーザと機械)の間でどのように情報をやり取りするか，ということを指す．
3. 前提知識: 学習対象に関する前提知識を指す．
4. 表現言語: 学習対象をどのように表現・記述するかを指す．
5. 学習アルゴリズム: 実際にどのような手続きを経て学習を行うかを指す．
6. 学習の精度・効率: 学習対象をどれほど「正しく」あるいは「効率よく」学習できるかを指す．

学習システムは，教師から学習対象に関する**例**を与えられ，それらの与えられた例を参考に学習目標に関する**仮説**を出力する．例とは学習目標に関するサンプルデータのことであり，多くの場合学習目標の集合に属するメンバが選ばれる．仮説とは学習目標に関する機械の推論結果であり，これが学習目標と等価となるように学習が進められる．

質問学習モデルでは，学習のプロトコルに関して，例を与えられるだけでなく生徒である機械から学習対象に関する質問を許す．与えられる例の集合だけを見る場合に比べ，生徒の求める情報を積極的に得ることができるため，学習を効率的に行うことが可能となる．

質問学習モデルにおいては，**所属性質問**と**等価性質問**がよく用いられる．本研究では，**包摂質問**という質問を導入する．

また質問学習では，やみくもに仮説を列挙するのではなく，現在の仮説を徐々に具体化（特殊化）したりあるいは抽象化（一般化）して，正しい仮説に近づける手法がよく用いられる．これにより，仮説どうしの持つ関係を用いて仮説を探索することができ，仮説探索の効率化に繋がる．この抽象化・一般化関係を問い合わせる質問が包摂質問である．

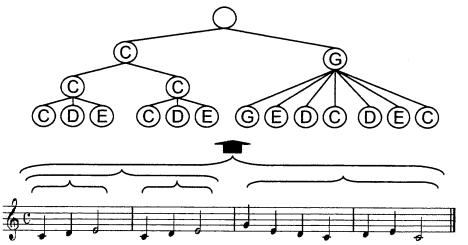

図1 楽曲の構造化

2.2 構造を持つ楽曲データ

本研究における楽曲データの構造とは，旋律の階層的なグルーピング構造と，音情報の部分的な抽象・具体化関係の情報を指す．この構造はGTTM[5]のタイムスパン木を参考にしている．

楽曲における音符は一つのノードによって表され，ノードが階層的に集まり木を形成し楽曲全体で一つの木構造をなす．本研究ではこのデータ構造を**MusicTree**と名づける．

MusicTreeは以下の条件を満たすラベル付き順序木である．

- 全てのノードは楽曲の中での対応する区間(タイムスパン)が保持されているラベルを持つ．
- ある葉でないノードが音の高さが保持されているラベルを持つならば，その子ノードは全て音の高さが保持されているラベルを持つ．
- ある葉でないノードのタイムスパンは子ノードのタイムスパンの総和に等しい．
- 祖先・子孫関係にない二つのノードのタイムスパンは重ならない．

シーケンシャル楽曲データに対して，旋律のグルーピング構造などを見出し，MusicTreeとして表現することを**楽曲の構造化**と呼ぶ．また，MusicTreeで表現される楽曲データを**MusicTreeデータ**と呼ぶ．図1に楽曲の構造化の例を示す．この例では，図中において五線譜で表現されるシーケンシャルデータをMusicTreeデータへと構造化している．まず，元のデータに階層的なグルーピング構造を見出し，それに基づいて高レベルのノー

ドを作成する．最終的に楽曲は，楽曲全体のタイムスパンを表す最上位のノードにまとめられている．また必要に応じて，各グルーピングにおいてそのグループを代表する音を親ノードの音の高さとして要約を行う．なお図中では各ノードのタイムスパンの長さの表記は省略している．

MusicTree データを対象とすることで，抽象的な，グルーピング情報しか持たないようなデータと，具体的な，実際に聞くことのできる音楽として成立するデータを一つのデータで表現できるので，仕様を具体化していき徐々に完成データに近づけるという操作を実現でき，現在の仮説を徐々に強めて目標に近づける，という質問学習の枠組みで扱うことが容易となる．

本研究では作曲という行為を，抽象的な MusicTree データを具体化していき，より精緻な MusicTree データにしていくことであると捉える．MusicTree データの具体化はノードの具体化の繰り返しによって実現される．**ノードの具体化**とはすなわち，あるノードに MusicTree の条件を満たすように複数の子ノードを付け加えることである．MusicTree の局所的な親子関係は具体化の適用事例を表していると考えられる．

また，具体化によってこのデータに対して二項関係を定義することができる．すなわち MusicTree データ M を具体化して得られたデータ M' に対して $M' \prec M$ と定義する．これは仮説の包摂関係に相当し，本研究ではこの関係を利用して学習の際に包摂質問を行う．

3 楽曲構造の学習

図 2 に学習アルゴリズム *MusicGen* の概要を示す．

ユーザは仕様を，後述する MusicTree データで表現し入力とする．仕様では楽曲が複数のグループに分けられ，グループをどの順で生成するかという情報が含まれる．$autogen(N, \tau)$ はノード N を受け取って具体化を行い，候補の MusicTree データ M を生成する関数である．また，τ は事前に決められた定数である．

以下ではアルゴリズムの詳細について説明を行う．

3.1 仕様の表現

ユーザは，学習の出発点となる MusicTree データを仕様としてシステムに与える．仕様は基本的に，構造を持った MusicTree データで表現される．仕様を表すためいくつかの追加表現を許している．

Procedure $MusicGen(T)$
T:ユーザが仕様として与える MusicTree データ．
ノード $N_1, \ldots, N_m$ がこの順で具体化される．
for($N = N_1$) to N_m **do**
  repeat
    $autogen(N,\tau)$ の結果 M_i を
    提示して包摂質問を行う．
    if Yes が返される **then**
      break.
    else
      MusicTree データの一部が返される．
      ユーザの評価情報を更新．
  foerever

$M_1, \ldots, M_m$ を T につないで得られる木 M を返す．

図 2　学習アルゴリズム

- **生成順の指定**
  楽曲への具体化を適用する順序を指定する．アルゴリズムはここで指定されたグループごとに質問を用いながら MusicTree データの生成を行う．MusicTree データの一部だけを生成したいという場合や，また，サビにあたるような主要な部分を最初に作ってその特徴を全体にも反映させるといったことを可能としている．

- **同一箇所の指定**
  楽曲の中で全く同一にしたい部分を指定できる．生成する際には片方のみ生成し，その具体化を他の同一箇所にも同様に適用する．

図 3 に仕様表現の例を示す．なお図中において × の後ろの数はそのノードのタイムスパンの長さを示す．この例では，曲全体をそれぞれ長さ８の４つの区間で構成し，最初の区間を要約すると音 C となり，左から２番目，１番目，３番目の順で生成を行い，最後の区間は左から２番目の区間と同じ部分木である，という仕様を表す．

3.2 ユーザへの質問

図 2 のアルゴリズムでは，質問は生成した候補の MusicTree データについてユーザに尋ねる，という所で用いる．この質問は，直感的に言うと，「この MusicTree データを具体化していくと，目標とする MusicTree データは得られるか？」という質問であり，質問学習モデルでは**包摂質問**と呼ばれる質問に相当する．すなわち，現在提示した途中

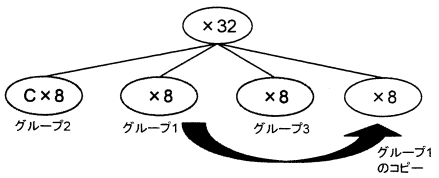

図 3　仕様表現の例

経過はユーザの意図に沿ったものになっているかということであり，ユーザは問題なければ Yes と答える．もしそうでなければ，システムが提示した MusicTree データに対して，ユーザの意図にそぐわない区間（タイムスパン）が含まれるということであるので，ユーザはその区間に相当する部分木を指定して質問への回答とする．システムは，指摘された部分木に含まれる具体化事例の評価を全て下げる．

このように，質問に対して回答を繰り返すことで徐々に，直接的には「ユーザの意図しない MusicTree データ生成しない」ようになるが，結果的に「ユーザの意図する MusicTree データが生成される」ようになる．質問学習を用いると，このような「楽曲が何だかおかしいという」情報を容易に与えることができ，ユーザの負担の軽減へ繋がる．

3.3　具体化の事例を用いた自動生成

MusicTree データの生成は，仕様データへ具体化を繰り返し適用していくことにより進められる．

具体化の対象となったノードに関して，事例から最も適切な具体化事例を選択してそれによりノードの具体化を行う．具体的には，事例ベースに登録されている具体化の適用事例との適合度を計算する（適合度の具体的な計算方法については後節で行う）．適合度が最大であった事例を選択し，その適合度が閾値以上であればそのノードへ具体化事例と同じ方法で具体化を行う．これを新しくできた子ノードに関しても再帰的に具体化を行う．逆にその適合度が閾値未満であれば，そのノードに対する具体化は失敗とし，その部分構造に関しては具体化をそこで停止させバックトラックを行う．なお，閾値はユーザが任意に設定できる．図 4 に自動生成のアルゴリズム *autogen* を示す．

3.4　事例ベースの構築

MusicTree データにおいて，作曲をデータの具体化の連続によって行われると捉えると，多くの事例においてどのような具体化が行われているかという情報を蓄積すれば，機械による楽曲の自動

function *autogen*(具体化対象のノード N, 閾値 τ)
　N と事例ベース内の具体化の事例との
　適合度 A を計算する．
　A の最大値 A_{max} と対応する具体化の事例 C を
　記録しておく．
　if　$A_{max} > t$　**then**
　　N に対して具体化事例 C と同じ方法で
　　子ノードを付加し，M とする．
　　新しく生成された子 M' それぞれに対して
　　$M' = autogen(M', \tau)$ とする．

　M を生成結果として返す．

図 4　具体化を用いた生成アルゴリズム

生成に役立てることができる．そのため，予め多数の MusicTree データを事例ベースとして登録しておく．なお事例ベースの作成は，グルーピングや要約関係を見出すといった高度な作業が伴われるので，現在は人手での作成に頼っている．

また，後述する，MusicTree データ内の具体化対象のノードの事例に対する適合度の計算で用いるため，局所的なの具体化事例だけでなく音列のシーケンスの情報を N-gram で標本化し，各組み合わせの出現回数を記録している．なお，現在の実装では $N = 3$ としている．

3.5　具体化事例と適合度

適合度とは，具体化対象のノードに対して，事例ベースに登録されているある具体化事例を適用するのがどれほど好ましいかを表す数値である．元の事例においてその具体化が使われていたケースと，具体化対象のノードが似ているほど適合度を高くする．

適合度 A は以下のパラメータ $a_1, \ldots, a_5$ を用いて定義される．

- **タイムスパンの長さに基づくパラメータ a_1**
  タイムスパンの長さが近いノードほど適合度を高くする．具体的には

$$a_1 = \frac{min(t_{tget}, t_{case})}{max(t_{tget}, t_{case})}$$

  (ただし t_{tget} は具体化対象のノードのタイムスパンの長さ，t_{case} は具体化事例のタイムスパンの長さ)

  とする．これにより，事例の中で抽象的なレベルにおける具体化事例は，同じように抽象的なレベルでの具体化に用いられることが多く，具体的な細部のフレーズを決めるような

具体化事例は同じように具体的なレベルで適用されることが多くなる．また，自動生成の際に延々と再帰的に具体化され音楽的に問題のあるデータが生成されるのを防いでいる．

- **音高に基づくパラメータ** a_2

具体化の対象となるノードと事例，それぞれのノードの音の高さが同じであるかどうかを比べる．具体的には，

$$a_2 = \begin{cases} 1 & (\text{音高が同じであるとき}) \\ c & (\text{音高が異なるとき}) \end{cases}$$

(c は $0 \leq c < 1$ を満たす定数)

とする．なお，結果的に具体化対象のノードと音の高さが異なる具体化事例が選択された場合，対象のノードに音の高さが合うように具体化後の音の高さをずらして具体化を行う．

- **グルーピング情報に基づくパラメータ** a_3

そのノードが，グルーピングにおいてどのような位置にいるかを判断するためのパラメータである．具体的には

$$a_3 = \frac{min(t_{tget1}, t_{case1})}{max(t_{tget1}, t_{case1})} \frac{min(t_{tget2}, t_{case2})}{max(t_{tget2}, t_{case2})}$$

(ただし t_{tget1}，t_{tget2} はそれぞれ具体化対象のノードの前後のノードのタイムスパン，t_{case1}，t_{case2} は具体化事例の前後のノードのタイムスパン)

とする．グルーピングにおいての位置は，対象となるノードに関してその前後にどれほどタイムスパンの長いノードが存在するかという点で判断する．前後それぞれのノードのタイムスパンが具体化の対象と事例ベース内の具体化事例とでどれほど近いかを見る．ここで言う前後のノードは，次のように探す:

前のノードを探す場合は，まず，対象となるノードが兄弟の中で一番前に (順序が先に) あるかどうかを調べる．もしそうであれば，対象ノードの親を新たな対象ノードとし，同様に兄弟の中で一番前にあるかどうかを調べる．もし違うならば (もしくは兄弟が存在しないならば) そこで終了して，対象ノード自身を前ノードとする．後ノードの場合も前後を入れ替えただけで同様である．図 5 に前後ノードの例を示す．図中の対象ノード A に関して説明をすると，ノード A は兄弟の中で一番前なので親に注目し，親は兄弟の中で一番前ではないので，そのノードを前ノードとする．また，ノード A は兄弟の中で一番後ろではないので自信が後ノードとなる．

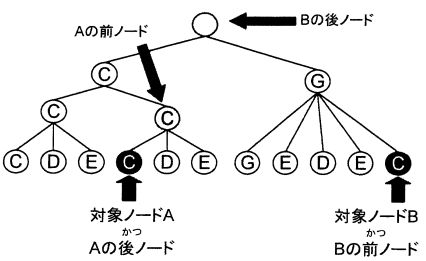

図 5　前後ノードの例

対象ノードがグループの切れ目の前後にあればその前後のノードはより大きなタイムスパンを持つことになる．すなわち，あるノードの前のノードが大きなタイムスパンを持つならば，そのノードはグループの最初のノードとしての性質が強いことを表す．逆に後のノードが大きなタイムスパンを持つならばそのノードはグループの終わりとしての性質が強いことを表す．

- **音高の推移に基づく適合度** a_4

楽曲を音符のシーケンシャルデータとして見たときに，ある音符は直前のいくつかの音符によって想起されるという仮定に基づくパラメータを，

$$a_4 = \prod_{i=1}^{N-1} (1 + freq_i)$$

(ただし $freq_i$ は，直前の i 個の音符の音高が出現したときに，続いて具体化事例に含まれる最初の $N-i$ 個の音符の音高が出現する頻度)

とする．頻度の計算には，N-gram による標本化を用い，どんな音列の組み合わせがどの程度出現したかを，MusicTree データを事例ベースに登録する際にあらかじめカウントしておく．

- **ユーザの評価に基づく適合度** a_5

同じ楽曲の中では，ユーザは同じ具体化事例を繰り返し使うことを好むという仮定に基づく適合度である．

$$a_5 = d^{n_{case}}.$$

(n_{case} は，その事例を拒否した回数．d は $0 < d < 1$ を満たす定数)

n_{case} の値は具体化事例ごとに存在し，ユーザに対する質問の結果更新される．

最終的な適合度 A は

$$A = a_1 \times a_2 \times a_3 \times a_4 \times a_5$$

と計算する．なお，適合度を計算する際に用いられる定数 c, d は，ユーザが任意に設定できる．

4 試用による評価

前章で提案したシステムを，VisualBasic 言語を用い WindowsOS 上で動作するように実装を行った．

さらに，このシステムを評価者５人に試用してもらい，評価と感想を得た．５人のうち２人は楽器の経験があったが，作曲の経験は全員なかった。試用の際の事例ベースは「チューリップ」「ちょうちょう」といった童謡９曲を元に人手で MusicTree データを作成し，構築を行った．事例ベースとしては規模が小さいが，最低限楽曲が生成できるレベルである．生成する楽曲は８小節の短い楽曲とし，４つのグループにわけ，システムへ入力する仕様データとした．

その結果，おおよそ１～５分程度の時間で楽曲構造の学習が終了し，作曲に慣れない人間や経験のない人間でも実用的な時間で楽曲が生成できることが確認できた．また生成された楽曲に関しても概ね好評であり，本アルゴリズムの有用性が検証できた．しかし，事例ベースに関しては実用上十分な数の事例が集まっているとは言い難く今後の充実化が望まれる．

5 おわりに

本研究では，作曲という行為を作曲させる側とする側の情報のインタラクションと捉え，質問学習という機械学習のモデルでモデル化し，その枠組みの中で動作する学習システムの設計を行った．また実際にシステムを試用してもらうことで，本アルゴリズムの評価を行った．

今後の課題としては事例ベースの構築が挙げられる．この学習アルゴリズムでは，生成される楽曲に現れるフレーズは事例ベースのみを用いて構成されるので，自然言語処理におけるコーパス同様，十分な数の事例を用意することが必要である．しかし，大規模な事例ベースの構築には人手による多大な労力が必要である．これを改善するためには，シーケンシャル楽曲データからグルーピング構造などの楽曲データ構造を効率よく発見するための手法が不可欠であるが．楽曲データからこのような構造を自動抽出しようとする研究も行われており [3]，これらの手法の応用は可能であると考えている．しかし，このような構造の抽出は専門知識を持った人間が行う場合でも結果が異なる場合もあるような曖昧な問題で，機械のみで全て行うことは難しく，そのためやはり最終的には人間の手作業が必要であろう．

また，ユーザへの質問の回答としてユーザは自分の気に入らない部分を指定して返すことしかできないが，ユーザの行う評価として見ると提示される候補にたいする否定しか行うことができず，自由度が少なく問題点の一つとなっている．事例ベースの中から適合度の上位の事例をユーザに選ばせるなど，ユーザへの負担が増大しない範囲での改善が必要である．

謝辞

本研究は一部，日本学術振興会 科学研究費 基盤研究 (B)19300046 の補助を受けている．

参考文献

[1] 安藤 大地, 丹治 信, 伊庭 斉志: EC を用いた作曲支援システムと作曲モデルの客観的評価手法, 情報処理学会研究報告, 2006-MUS-68, pp.29-34, 2006.

[2] Angluin, D.: Queries and Concept Learning, Machine Learning, 2, pp.319-342 1988.

[3] Masatoshi, H., Keiji H., Satoshi T.: Implementing "A Generating Theory of Tonal Music", Jornal of New Music Reserch (JNMR), Vol. 35, No. 4, pp. 249-277, 2006.

[4] 平田 圭二, 青柳 龍也: バービーブン: 音符レベルでユーザ意図を把握して編曲を行う事例ベースシステム, 情報処理学会研究報告 2000-MUS-37, pp.17-23, 2000.

[5] Lerdahl, F., and Jackendoff, R.: A Generative Theory of Tonal Music, MIT Press, 1983.

[6] 榊原 康文, 小林 聡, 横森 貴: 計算論的学習, 培風館, 2001.

[7] Shapiro, E. Y.: Inductive Inference of Theories from Facts, Technical Report 192, Yale University 1981. Also in J.-L. Lassez and G. D. Plotkin. (eds.) Conmputaional Logic. The MIT Press, pp. 199-254, 知識の帰納的推論, 淵一博 監修, 有川節夫 (訳)