3N-08

Variational Auto Encoder を用いたメロディとコードのモーフィング

村田 叡 † 坂東 宜昭 ‡ 糸山 克寿 ‡ 吉井 和佳 ‡
† 京都大学 工学部情報学科 ‡ 京都大学 大学院情報学研究科 知能情報学専攻

1. はじめに

ソフトウェア技術の進歩に従い, 音楽の楽しみ方も多岐に渡り, 素人でも作曲を行えるようになった. ユーザーに音楽経験があれば DTM ソフトウェアを用いてピアノロールを配置し楽曲を制作できる. 音楽知識が無くとも自動作曲ソフトウェアを用いて, ユーザーが指定したジャンル・曲調・雰囲気から自動で楽曲を生成できる. しかし, 従来の自動作曲では大まかな情報しか指定できず, ユーザーの意図を完全に反映できないという問題がある.

本稿では自動作曲のための楽曲のモーフィングについて述べる. モーフィングは, ある楽曲を別の楽曲に近づける操作であり, ある楽曲に別の楽曲のニュアンスを付与できる. また一から作曲するよりも容易であり, 自動作曲よりもユーザーの意図に沿った楽曲が生成できる. 従来のモーフィング手法として平田らによる生成音楽理論 (GTTM) に基づいた手法 [1] が存在し, 文法木の構造が近い楽曲同士であればメロディ同士のモーフィングができる. 他にも Adam ら [2] は, Variational Auto Encoder (VAE) [3] を用いて, 楽曲の文法構造に関わらずメロディ・ドラム・バスの個別のモーフィングを実現した.

本研究では, VAE を用い, メロディに加えてコードも同時に学習する. 音楽においてメロディとコードとその進行は密接な関係にあるため, コードもモーフィングできればユーザーの作曲体験はより豊かになる.

2. VAE を用いた楽曲モーフィング

本研究では, 楽曲の特徴を表す低次元の連続潜在空間を仮定し, 潜在空間上の補完により図 1 のようにモーフィングを実現する. 潜在空間と楽曲との関係は VAE に基づく確率的生成モデルとして定式化する. ある楽譜に対応する潜在変数は, その事後分布から推定できる.

2.1 生成モデル

楽曲 $\boldsymbol{x}$ はメロディ系列とコード系列により表現する. ただし $\boldsymbol{x}$ は 16 分音符単位で長さ T とし, 本稿では $T = 64$ とした. 簡単のためメロディ系列は音高系列とオンセット系列で表現し, 休符を考慮しない. ある時刻の音高はメロディとして取りうる音高数 P を用いて $p_t \in \{1, ..., P\}$, オンセットは二値 $h_t \in \{0, 1\}$ として表す. コードは取りうるコードの種類数 C を用いて無音状態も含めて考え $c_t \in \{0, ..., C\}$ と表す. 以上より, 楽曲 $\boldsymbol{x}$ の次元 N として, $\boldsymbol{x} = \{\boldsymbol{p}, \boldsymbol{h}, \boldsymbol{c}\} \in \mathbb{R}^N$ と表現される.

コードとメロディの関係や音楽の繰り返し構造といった楽曲の特徴を表す D 次元の潜在変数 $\boldsymbol{z}$ を仮定する. これが具体的にどのような特徴を表すかは VAE を用いて学習する. 従来の VAE と同じように $\boldsymbol{z}$ は多変量標準ガウス分布に従うとする. この分布は連続空間であるため線形補間を行えることに注意する. x_t の各要素である p_t は

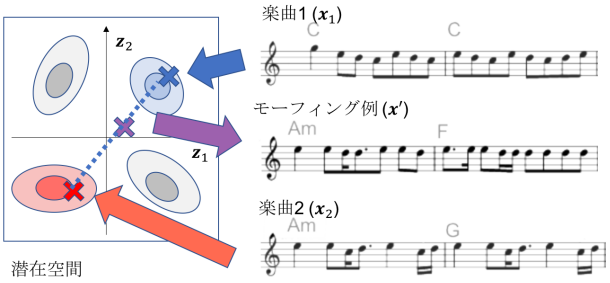

図 1: 楽曲のモーフィング概念図

Categorical($\pi_t^p(\boldsymbol{z})$) に. h_t は, Bernoulli($\pi_t^h(\boldsymbol{z})$) に, c_t は Categorical($\pi_t^c(\boldsymbol{z})$) に従うとする. ここで $\pi_t^p(\boldsymbol{z}) : D \to \{1, ..., P\}, \pi_t^h(\boldsymbol{z}) : D \to \{0, 1\}, \pi_t^c(\boldsymbol{z}) : D \to \{0, ..., C\}$ は, $\boldsymbol{z}$ との関係を表す非線形関数で, VAE を学習して得られる.

2.2 VAE と学習

K サンプルの学習データ $\boldsymbol{X} \in \mathbb{R}^{K \times N}$ と, それに対応する潜在変数を $\boldsymbol{Z} \in \mathbb{R}^{K \times D}$ として周辺尤度 $p(\boldsymbol{X})$ を最大にする $p(\boldsymbol{X}|\boldsymbol{Z})$ の分布を求めたいが, 解析的な計算は通常困難である.

$$\arg\max_{p(\boldsymbol{X}|\boldsymbol{Z})} p(\boldsymbol{X}) = \arg\max_{p(\boldsymbol{X}|\boldsymbol{Z})} \prod_d \int p(\boldsymbol{X}|\boldsymbol{Z}) p(\boldsymbol{Z}) d\boldsymbol{Z} \quad (1)$$

そこで $\boldsymbol{Z}$ の事後分布を変分ベイズ法を用いて下式のように変分事後分布で近似する.

$$p(\boldsymbol{Z}|\boldsymbol{X}) \approx q(\boldsymbol{Z}) = \prod_{k,d} q(z_{kd}) = \prod_{k,d} \mathcal{N}(\mu_d^{\boldsymbol{z}}(\boldsymbol{x}_k), \sigma_d^{\boldsymbol{z}}(\boldsymbol{x}_k)) \quad (2)$$

ここで, $\mu_d^{\boldsymbol{z}}(\boldsymbol{x}), \sigma_d^{\boldsymbol{z}}(\boldsymbol{x})$ は変分事後分布を表す一次元ガウス分布の平均と分散のパラメータである. 対数周辺尤度 $\log p(\boldsymbol{X})$ の変分下限が下式で求まる.

$$\begin{aligned} \log p(\boldsymbol{X}) &\geq \int q(\boldsymbol{Z}) \log \frac{p(\boldsymbol{X}|\boldsymbol{Z}) p(\boldsymbol{Z})}{q(\boldsymbol{Z})} d\boldsymbol{Z} \\ &= \mathbb{KL}[q(\boldsymbol{Z})|p(\boldsymbol{Z})] + \mathbb{E}_q[\log p(\boldsymbol{X}|\boldsymbol{Z})] \quad (3) \end{aligned}$$

ここで $\mathbb{KL}[*|*]$ は Kullback-Leibler 擬距離を表す. VAE を用いることでこの下限が最大となるように $q(\boldsymbol{Z})$ と $p(\boldsymbol{X}|\boldsymbol{Z})$ を表すニューラルネットワークを学習できる.

3. 実験

実際に上記のモデルで学習を行い, メロディとコードのモーフィングの生成結果を確認した.

3.1 実験条件

音高数 $P = 50$ とし, コード数 C は ルート音 12 種類に対し { Major, Miner } を考えた $12 \times 2 = 24$ とし, No Chord の状態も含めて学習する. コーパスは, RWC Music DataBase[4] 及び, Nottingham Music DataBase[5] を用い

Melody and Chord Morphing using Variational AutoEncoder : *Satoshi Murata, Yoshiaki Bando, Katsutoshi Itoyama, and Kazuyoshi Yoshii (Kyoto Univ.)*

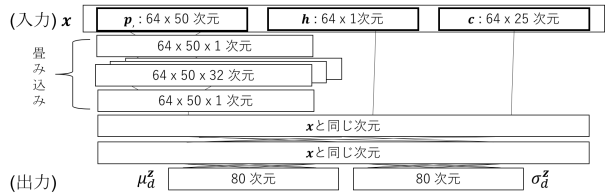

図 2: $q(\boldsymbol{z})$ を学習するネットワーク

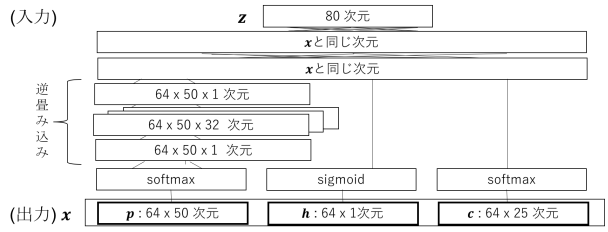

図 3: $p(\boldsymbol{x}|\boldsymbol{z})$ を学習するネットワーク

た. 調は事前に C Major または C Minor に移調した. 各コーパスのデータには三拍子の楽曲が存在するため, またデータ数を増やすためにも, 各曲では幅 4 分音符ずつずらして使用した. 三連符や 32 分音符などの 16 分音符の分解能では扱えないデータや, ディミニッシュコードやセブンスコードなどの想定していないコードを含むデータも多く存在するが, それらを含むデータは学習データから除外した.

3.2 ネットワークの構造

2.2 節で述べた VAE を構成するネットワークの構造について述べる (図 2, 図 3). $q(\boldsymbol{z})$ を表す層は音高 $\boldsymbol{p}$ を 3 層の CNN を経たものを $\boldsymbol{h}$, $\boldsymbol{c}$ と結合し, $\boldsymbol{x}$ と同次元のユニット数を持つ全結合層 2 層を通して学習する. $p(\boldsymbol{x}|\boldsymbol{z})$ を表す層は $\boldsymbol{x}$ と同次元の全結合層 2 層を通し, 音高 $\boldsymbol{p}$ のみ 3 層の CNN 層を経て学習する. 学習が行われやすいよう, 離散分布となる $\boldsymbol{p}$,$\boldsymbol{c}$ には softmax 関数の層を, ベルヌーイ分布となる $\boldsymbol{h}$ には sigmoid 関数の層を加えた.

CNN 層ではゼロパディングにより層の前後で次元数は一定にし, カーネルサイズは全て音高方向にも位置方向にも 4 とする. 過学習の抑制と学習の高速化のために各層に Batch Normalization 層を挿入した. 最適化手法には Adam ($\alpha = 0.0001$, 他パラメータは提案論文 [6] に従う) を用い, 各層の間の活性化関数には LeakyReLU ($a = 0.2$) を用いた. バッチサイズは 100 とし,92030 データを 100 エポック学習した.

3.3 考察

図 4 の譜面は, 前述コーパス内の学習データに含まれていないデータをモーフィングしたものである. 最上段及び最下段の譜面がモーフィング対象となる 2 つの楽曲 $\boldsymbol{x}_1$ および $\boldsymbol{x}_2$ であり, 中間の譜面は全て VAE により自動で生成された楽曲である. モーフィング途中の楽曲には例えば図 4 の 3 段目譜面の波線部分に新たな局所的な繰り返し構造が見られるように, 音楽的な構造を無視せずに変化していることが分かる. また, 図 5 のようにコードの情報を与えずに再構成した場合, 近いメロディを保ったまま自動でコードが付与されるという効果が確認された. これはコーパス中には常にコードが存在していたためだと考えられる. 図 6 のようにほとんどが 16 分音符で

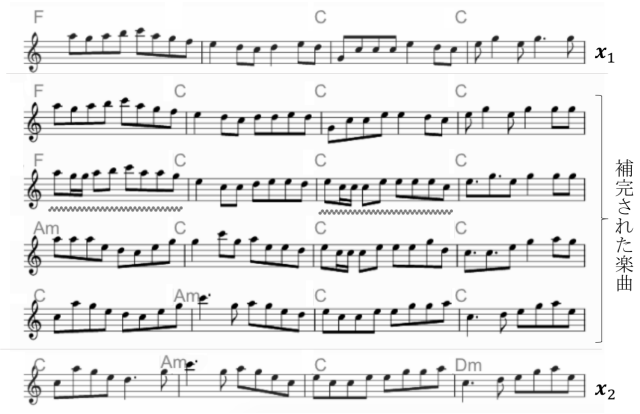

図 4: メロディとコードを含む 2 楽曲間のモーフィング例

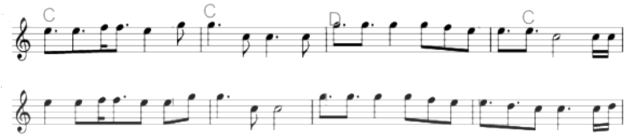

図 5: 再構成によりコードが付与される例

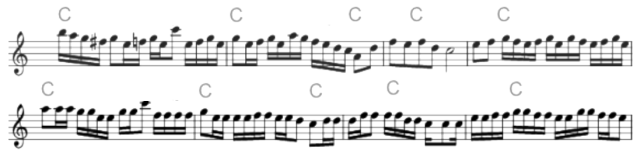

図 6: 再構成の失敗例 (上段:元の楽曲, 下段:再構成楽曲)

構成される楽曲は上手く再構成できなかったという課題がある. これは実験ではデータ数が少ないためやや過学習を起こしているからだと考えられる.

4. おわりに

本稿では VAE を用いてメロディとコードを持つ二楽曲間を自動でモーフィングする方法を述べた. 実験により異なる 2 楽曲間のメロディ・コード共にモーフィングできていることが確認された. 楽曲データに偏りがあったので今後はより大きなデータセットによる学習を試みたい. また, ユーザーがモーフィングをより自由に行えるよう, モーフィングの前後でコードまたはメロディの片方を固定するモデルへの拡張を考えている.

謝辞 本研究の一部は、JSPS 科研費 26700020, 16H01744 および JST ACCEL No. JPMJAC1602 の支援を受けた.

参考文献

[1] Keiji Hirata *et al.*: "Melodic Morphing Algorithm in Formalism," *MCM*,pp. 338–341 2011.

[2] Adam Roberts *et al.*: "Hierarchical Variational Autoencoders for Music," *NIPS Workshop*, 2017.

[3] Diederik P Kingma *et al.*: "Auto-encoding Variational Bayes," *arXiv preprint*, arXiv:1312.6114, 2013.

[4] Masataka Goto *et al.*: "RWC Music Database: Popular, Classical and Jazz Music Databases," *ISMIR*, pp. 287–288, 2002.

[5] James Allwright: "ABC version of the nottingham music database," 2003. URL: `http://abc.sourceforge.net/NMD/`

[6] Diederik P Kingma *et al.*: "Adam: A Method for Stochastic Optimization," *arXiv preprint*, arXiv:1412.6980, 2014.