平成 27 年度　特別研究報告書

MIDI データと自己組織化マップを組み合わせた演奏曲調分類の研究

龍谷大学　理工学部　情報メディア学科

T120425　垣田 詩鶴香

指導教員　三好 力　教授

内容梗概

一般的に人が何らかの用途で楽曲を選ぶ際は，歌手・演奏者・作曲者・ジャンル・曲調等の特徴を利用していると考えられる．あるお店に行くとそのお店の雰囲気に合った曲調の楽曲を流していることが多いように感じる．また，楽器を練習する際の練習曲を選ぶ際にも好みの曲調で選んだり，反対に技術や表現の幅囲を広げるためにそれまでと異なる曲調の楽曲を選んだりすることがあるだろう．このように曲調は選曲の上で重要度が高いと考えられるが，ジャンルや曲調は感性を用いた曖昧な分類であるため，歌手や演奏者，作曲者で選別することに比べて，それらで選別することは容易ではない．

そこで本研究では，自己組織化マップ(Self-Organizing Map)を用いて MIDI 形式のデータ(MIDI データ)の曲調分類を試みた．具体的には，ジャンル・曲調の異なる多数の楽曲の MIDI データをバイナリデータとして取得し，そこから 16 進数 2 桁の 0x00 から 0x7F の範囲の 128 次元，もしくは 16 進数 1 桁の 16 次元のヒストグラムを作成する．最後にこのヒストグラムを学習データとして SOM を作成し，楽曲とジャンル・曲調の関係を調べるという実験を行った．

目次

1. 緒論

一般的に，人が何らかの用途で楽曲を選ぶ際は，歌手・演奏者・作曲者・ジャンル・曲調等の特徴を利用していると考えられる. それらの特徴を単一もしくは複数個用いて，数多ある楽曲から一つの曲を選び取る.

楽曲の曲調による分類に注目すると，たとえば，街中を歩いているとあちこちのお店から音楽が聞こえてくるが，お店の雰囲気に合った曲調の音楽を流していることが多いように感じる. 落ち着いた雰囲気のレストランでは穏やかなクラシックが流れていたり，賑やかな居酒屋ではアップテンポの流行曲が流れているだろう．他にも，のんびりしたい時にゆったりとした曲，元気になりたい時や目を覚ましたい時にアップテンポの曲など，気分や用途に応じて曲調で選曲することもあるだろう.

曲調分類は音楽を聴く際に使うことも出来るが，楽器を練習するための練習曲を選ぶ際にも有用である．従来練習曲は既知の曲や作曲者・演奏者などから選び，有名な曲や有名な作曲家など, 偏った選曲になることが多かった. しかし曲調分類が可能になると，曲調が同じだが知らない曲からも選ぶことが出来，より好みの曲や作曲家を新しく見つけることの出来る可能性が増える．また，演奏した情報を入力に用いると，演奏曲と同じような曲調の曲や，反対に異なる曲調の曲にどのような曲があるかを簡単に調べることが出来る．練習曲に同じような曲調の曲を選べば，より技術力や表現力を深めることが出来るし，異なる曲調の曲を選べば技術や表現の幅を広げることが出来ると考えられる．また，自動分類出来るなら楽譜の難易度も含めた分類結果を表示させることが期待できるため，演奏者と全く違うレベルの楽譜を選ぶこともない．人の手で楽曲を選曲する際には楽譜の難易度は実際に楽譜を見てみないと判断出来ないため，一度楽曲を選んでも楽譜の難易度が演奏者のレベルとかけ離れていると，選曲は一からやり直さなければならない．つまり，自動分類を用いて選曲をすれば，選曲の際の手間が一気に省けるのである．選曲にかかる時間を練習の時間に回すことが出来るため，より効率の良い練習が可能になることが期待される.

このように曲調は選曲の上で重要度が高いと考えられるが，歌手や演奏者，作曲者で選別することに比べて，ジャンルや曲調は感性による分類であるため，それらで選別するのは容易ではない．そこで本研究では自己組織化マップ(Self-Organizing Map：以下 SOM とする)と MIDI 形式の音楽データ(以下 MIDI データとする)を用いた曲調分類を検討した.

MIDI データは音響波形情報(以下波形情報とする)と比べてデータ量が少なく，また，MIDI に対応した電子楽器，いわゆる MIDI 楽器を演奏してその演奏情報を計算機に入力するとこの形式で保存されるため，処理が容易である.

波形情報から音高や音の長さを得るためにはフーリエ変換などの処理が必要になるため，全ての処理を行うには多くの時間が必要になり，計算量が大きくなる．また，演奏情報は MIDI データよりもピッチのずれや音の長さにばらつきが大きい傾向がある．一方 MIDI データの場合学習用データには楽譜とほぼ変わらない情報を用いるため，正規

化の必要はなく，音の高さや音の長さは格納されているそのままのデータを使うことが出来る．また，演奏曲の入力データも波形情報を用いるよりも MIDI 楽器の演奏情報から得た MIDI データを用いた方がフーリエ変換等の処理が不要であるため処理が容易である．これらの理由から，本研究では波形情報ではなく MIDI データを用いる．

音楽におけるジャンルや曲調とは, 定義付けが曖昧である. 本研究では, “クラシック”, “ロック”, “ポップス”, ” タンゴ”, ” サンバ” などの音楽の様式や形式のことをジャンル, “穏やか”, “勇ましい”, “華やか”, “なめらか”, “迫力のある” などの楽曲の調子や雰囲気のことを曲調と定義する．ジャンルと曲調は異なる次元として扱うため，必ずしも連動する訳ではない．例えば，同じクラシックのジャンルにも穏やかな楽曲や迫力のある曲があり，ポップスの楽曲にも穏やかな楽曲や意気揚々とした楽曲が存在する．本研究ではジャンルを超えた曲調ごとの分類に対する分類手法の提案を行う．

2.基本事項

2.1.1.SOM について

SOM とは自己組織化マップの略称であり, データを与えるだけで機械が学習してくれる，教師なし学習の手法の一つである．教師なし学習であるため，特徴量に重みを付けたりラベリングの必要がない．他にも，多次元のデータを二次元や三次元などの低次元に落とすことが出来，地図上の距離で類似度を推定することが可能であるため，わかりにくい多次元のデータを低次元に落として直感的にわかりやすい形で出力出来ることが特徴である．一般的に出力結果は二次元もしくは三次元上に四角形(直方体)や六角形(正十二面体)のハチの巣の形状で表される．汎化能力が高く結果がわかりやすいため容易に曲調で選別するのに適している．

2.1.2.SOM の学習アルゴリズム

コホネンは生物の神経細胞, 主に人間の脳の情報処理の仕方を以下の簡単な式で表した．

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{2.1}$$

この式は, 次のような意味を持っている. 神経細胞(ノード) i が時刻 t で処理している情報処理能力を$m_i(t)$とする．外部から入力信号$x(t)$が入ってきた時, 細胞はこの入力信号を学習して次の時刻には入力信号により近い情報処理能力$m_i(t+1)$を持つようになる. この時, $x(t)$ が n 次元の入力ベクトルである場合, 参照ベクトルとも呼ばれる$m_i(t)$もまた同じ n 次元の要素を持つ．そして, $h_{ci}(t)$は学習率係数を含めた近傍関数を意味する．ここで, $t = 0, 1, 2,$ ...は離散時間座標である.

SOM において出力競合層のベクトルは参照ベクトル$m_i(t)$で表され, 入力層の次元に合わせて n 個の要素を持っている.

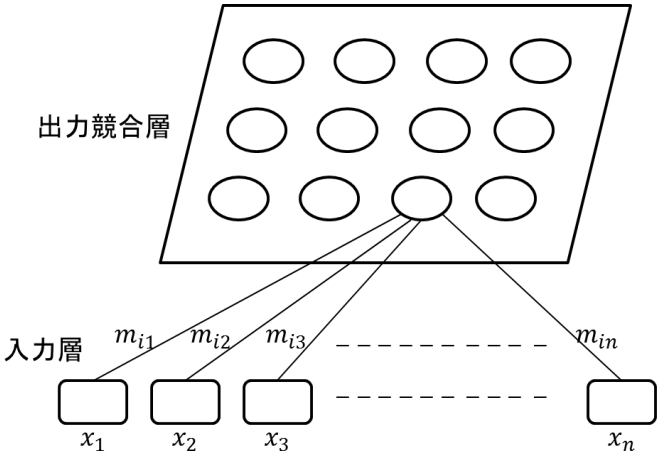

図 2.1:SOM の構造

自己組織化マップの学習は次のように行われる．入力ベクトル x(t)はある測度, 例えばユークリッド距離 $|x(t) - m_i(t)|$ を最小にするノード i を探し, それに添え字 C をつける.

$$|x(t) - m_c(t)| = min|x(t) - m_i(t)| \tag{2.2}$$

式(2.2)で決められた参照ベクトル$m_c(t)$を持つノード C を勝者ノードと呼ぶ. 式(2.1)と式(2.2)の時の学習の状態を図 2.2, 図 2.3 に示す. まず入力ベクトルが提示されると, その入力ベクトルに一番近いノードが勝者ノードとなる. そして勝者ノードの周囲のノード, つまり図 2.3 の円で囲われている領域を近傍領域と定義する. たとえば基本ノードの配列を六角形の形をした場合は周囲 6 個の六角形が近傍領域ということになる. この時, 領域の形は正方形でもよい. その近傍領域内の全てのノードは入力ベクトルを学習し, 式(2.1)に従って入力ベクトルの方向へ修正される. 近傍領域の範囲は最初は広く指定しておき, 学習が進行するにつれて近傍領域を狭めていく. この学習を繰り返し行っていく.

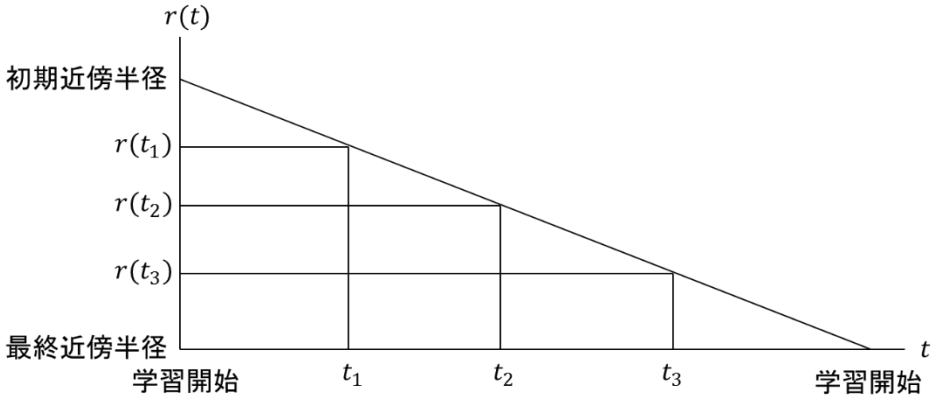

図 2.2:近傍領域を表す円の半径のグラフ

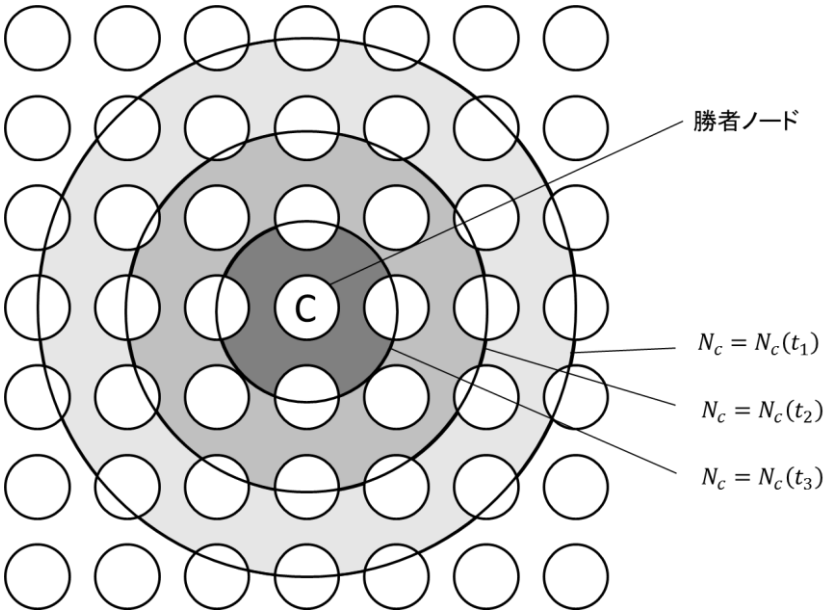

図 2.3:SOM の学習の繰り返しにおける近傍領域の変化

式(2.1)において, $h_{ci}(t)$は学習率係数$\alpha(t)$と近傍関数$h(d,\ t)$により以下の式のように表現できる.

$$h_{ci}(t) = \alpha(t) * h(d,\ t) \tag{2.3}$$

近傍領域は，学習される近傍領域を指定する関数である．近傍領域は，出力競合層の勝者ノードの近傍を意味し，学習によって参照ベクトルが更新される領域である．学習をする際，最初は近傍領域をその範囲を大きくとり，学習が進行するとともに徐々にその領域を狭める．近傍領域を減少させる関数には，線形型とステップ型が存在する．ここではよく使用されるガウス型近傍関数について説明する．この時，d は勝者ノードから参照ベクトルまでの距離，$k\big(r(t)\big)$は学習時間 t のときの近傍領域の最大距離で，ガウス関数の波幅の係数である．

$$\big(d,\ t\big) = exp\left(-d^2/k\left(r(t)\right)^2\right) \tag{2.4}$$

また，d は式(2.5)で表せる．

$$d^2 = (x - a_i)^2 + (y - b_i)^2 \tag{2.5}$$

ここで，$\big(a_i,\ b_i\big)$は勝者ノードの位置，(x, y)は半径$r(t)$から成る円形領域の内側にあるノードの位置を示す．

近傍領域の半径$r(t)$は，学習の始めでは大きく，学習が進行すると小さくなっていく．また，半径$r(t)$の円領域に含まれる範囲のノードや参照ベクトルは，指数関数によって決められる重みを持って学習する．すなわち，勝者ノードに近いほどその類似性が大きくなるように学習され，勝者ノードから遠ざかるほど，その類似性が小さくなるように学習される．そして，半径$r(t)$の範囲外のノードは学習が行われないようになる．

したがって，式(2.1)によって学習している間，近傍領域N_c内のノードに関しては$h_{ci}(t) = \alpha(t) * h\big(d,\ t\big)$，近傍領域$N_c$外のノードに関しては，$h_{ci}(t)= 0$ である．つまり，近傍領域の外側の領域では学習は行われない．

以上より，近傍関係を以下の式で表す．

$$\begin{aligned} h_{ci}(t) &= \alpha(t) * h\big(d,\ t\big) \quad (i \in N_c) \\ h_{ci}(t) &= 0 \quad (i \notin N_c) \end{aligned} \tag{2.6}$$

この時，$\alpha(t)$を学習率係数と呼び，$0 < \alpha(t) < 1$の値を持つ．$\alpha(t)$とN_cの大きさはどちらも学習時間が経つにつれて普通，単調減少させる．例えば，$\alpha(t)$であれば次の式で定義してもよい．

$$\alpha(t) = \alpha_0(1 - t/T) \tag{2.7}$$

ここで，a_0はaの初期値であり，普通 0.2～0.5 の値を選ぶ．T は行われるべき学習での予定された全更新学習回数である．ただし，式(2.1)中，比例係数の$a(t)$は学習の最初では大きな値をとるように設定し，学習の進行とともにその値を減少させるようにする．また，近傍領域$N_c = N_c(t)$も式(2.6)と同様に減らしていってもよい．つまり，

$$N_c(t) = N_c(0)(1 - t/T) \tag{2.8}$$

ここで，$N_c(0)$は初期値である．

コホネンの自己組織化マップアルゴリズムは，整理すると以下の手順で行う．

1. 出力層にノードを配置し，それぞれの持つ参照ベクトルを乱数で初期化する．
2. 入力ベクトルに最も近い(類似している)競合層での参照ベクトルを探し，これを勝者ノードとする．
3. この勝者ノード及びその近傍内のノードを式(2.1)に従って更新する．また，学習が進むにつれて近傍領域を狭め，学習率係数の値も式(2.4)のように減少させていく．

自己組織化マップの学習を行うために，学習パラメータを設定する必要がある．ここに主要な学習パラメータとその説明を示す．

学習回数：競合層に対して行う学習の回数を設定する

学習係数：一回の学習でノードをどれだけ更新するか設定する

近傍領域：学習に影響を与える範囲の初期値を設定する

2.2.MIDI について

MIDI とは Musical Instrument Digital Interface の略称であり，日本の MIDI 規格協議会と国際団体の MIDI Manufactures Association により策定された電子楽器の演奏データを機器間でデジタル転送するための世界共通規格である．MIDI データは 1 つの音に対して，発音する際のノート・オン・メッセージと消音する際のノート・オフ・メッセージの 2 種類の MIDI メッセージが存在する[1]．それぞれステータス・バイトと 2 つのデータ・バイトの 3 バイトの MIDI メッセージから構成される．MIDI データは，ステータス・バイトとデータ・バイトを見分けるために，2 進数 8 ビットの内の最初の 1 ビットを変化させている．最初の 1 ビットが 0 ならその MIDI メッセージはデータ・バイトであり，1 ならステータス・バイトとなっている．つまり，ステータス・バイトとデータ・バイトを区別するためには最初に 1 ビットに着目すれば良い．16 進数 2 桁でいうと，0x00 から 0x7F がデータ・バイト，0x80 から 0xFF がステータス・バイトとなる．ここから MIDI データの音は高さ，長さ，強さの取り得る範囲が 10 進数でいう 0 から 127 の範囲ということがわかる．ノート・オン・メッセージはステータス・バイトとデータ・バイトが 2 バイト存在するが，ステータス・バイトは先頭 4 ビットが 1001 で，後ろ 4 ビットがチャンネル・ナンバーを表している．MIDI には 1～16 の MIDI チャンネルが存在し，それぞれの番号のことをチャンネル・ナンバーという．0～15 ではなく 1～16 であるため，4 ビットが表す数に 1 を足した数がチャンネル・ナンバーであることに注意しなければならない．ノート・オン・メッセージのデータ・バイトは 1 つ目のデータ・バイトがノート・ナンバー，つまり音の高さを表しており，2 つ目のデータ・バイトはノート・オン・ベロシティ，つまり音の強さを表している．

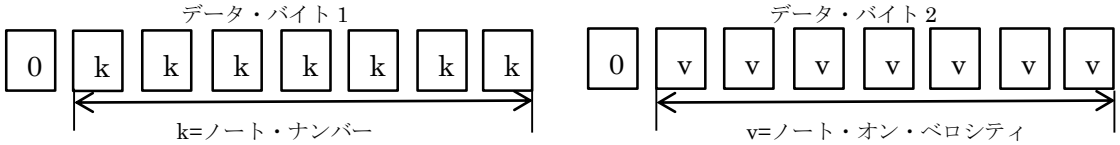

図 2.4：ノート・オン・メッセージのデータ・バイト

ノート・オフ・メッセージのステータス・バイトは先頭の 4 ビットが 1000 もしくは 1001 であり，後半の 4 ビットはノート・オン・メッセージと同様にチャンネル・ナンバーを表している．データ・バイトも同様に 1 つ目はノート・ナンバーであり，2 つ目はノート・オフ・ベロシティ，つまり音の強さを表している．

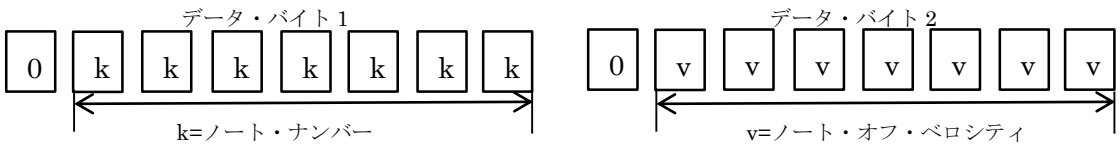

図 2.5：ノート・オフ・メッセージのデータ・バイト

MIDI にはデルタタイムという特有の時間間隔が存在する．デルタタイムの単位はチックといい，上述の通り 16 進数 2 桁で表され，10 進数でいう 0～127 の値をとる．デルタタイムは通常，メッセージの先頭にあり，1 つ前のメッセージの何チック後に後ろに続くメッセージを行うかという待ち時間を表す．ある音のノート・オン・メッセージか

らその音のノート・オフ・メッセージまでに存在するデルタタイムを足し合わせることによって音の長さを得ることが出来る.
MIDI データの全体としては，ヘッダ部とトラックデータ部からなり，トラックデータ部は前述のステータス・バイト(システムコマンド)，データ・バイト(音の高さ，音の長さ，音の強さ)の各情報単位を 2 進数 8 ビット，つまり 16 進数 2 桁を組としたデータ列からなる.

3.既存技術

本研究の目的である曲調分類に用いる曲調は，感性の問題であり，聴く人によって楽曲に対する印象が異なる．そのため，歌手・演奏者・作曲者・ジャンルごとの分類と比較しても，曲調に関する分類手法は一般的ではない．また,「曲調解析に基づく音楽分析法の提案」[2]は楽曲から音声信号を細分割し，音響特徴を抽出する．その後音響特徴を正規化し，サポートベクターマシン(以下 SVM とする)に入力する．SVM の多クラス分類法を用いて音楽を分類する，という分類法を提案した研究である．これは曲調を扱った音楽分類の提案ではあるものの,“クラシックの雰囲気が感じられるジャズ楽曲”というように曲調を判定に用いたジャンルごとの分類であり，ジャンルを超えた曲調に対する分類ではない．また，円グラフとして楽曲のジャンルを可視化するため，単一楽曲に対しての分類であり，楽曲間の比較や類似度を推測することは出来ない．さらに SVM は教師あり学習の手法であるため，学習用データにはラベリングの必要があり，音声信号の細分割，音響特徴の抽出・正規化，SVM と処理に多くの時間がかかる．そのため，本研究ではジャンルを超えた曲調分類手法を提案する．

4.提案手法

一般的に，人が何らかの用途で楽曲を選ぶ際は，歌手・演奏者・作曲者・ジャンル・曲調等の特徴を利用していると考えられる. それらの特徴を単一もしくは複数個用いて，数多ある楽曲から一つの曲を選び取るが，歌手や演奏者，作曲者で選別することに比べて, ジャンルや曲調は感性による分類であるため, それらで選別するのは容易ではない. そのため，本研究では SOM を用いた，MIDI データについて曲調分類を行う手法について提案する．具体的な入力としては，求める曲調の楽曲の MIDI データから抽出した情報，出力としては 2 次元マップ状の SOM となる．MIDI データから音の高さ，長さ，強さの特徴量を抽出し，その情報を入力として SOM を作成し，曲調ごとに分類する．

まず学習用にあらゆる曲調の MIDI データを用意する．その MIDI データから SOM を作成する．聴取用に用いる場合は，求める曲調の楽曲の MIDI データを用意し，作成した SOM に入力する．似た曲調の楽曲を探す際には SOM 上にて，入力した楽曲の近くに表示された楽曲を選び，異なる曲調の楽曲を探す際には近くでない場所に表示された楽曲を選び取る．練習曲の選定の際に用いる場合は，MIDI 対応楽器を演奏し，その演奏情報を情報端末に入力した後に SOM に入力する．その後は聴取用と同様に，似た曲調の楽曲を探す際には SOM 上にて，入力した演奏情報の近くに表示された楽曲を選び, 異なる曲調の楽曲を探す際には近くでない場所に表示された楽曲を選び取れば良い.

この手法において最も重要な点は MIDI データから SOM を作成する際に何を入力とするか，という点である．そのため，本報告では入力とするデータについての検討と実験について述べる.

5.実験

5.1.実験 1

MIDI データから得られるメロディー全ての楽器の音の高さや長さや強さをまとめた音の流れと，前述のジャンルや曲調の間に関係性が見られるかどうかを確認する実験を行った．本実験は，音の高さ，長さ，強さでジャンルや曲調を分類出来るという仮説を証明するために行ったものである．

仮説を確認するために多くのジャンルの曲の音の高さ，長さ，強さの部分を取り出しヒストグラムを作成した．音の高さ，長さ，強さはそれぞれ同じ値域で表されるため，128 次元の分布が 3 つ得られるが，それらの分布を重ねても特徴は失われないと考えた．具体的には SOM に学習させ，マップ上の距離から示された曲間の類似性と曲のジャンルを比較した．

5.1.1.実験 1 のデータ

実験には様々なジャンルと曲調の楽曲 102 曲を用意し，それらを入力ベクトルとして SOM を作成した．SOM は 10×10 のノードで出力した．なお，今回用いた楽曲は RWC 研究用音楽データベースから得た 102 曲であり，ジャンルはデータベース上に定義されている，ポップス，バラード，ロック，ヘビーメタル，ヒップホップ，ハウス，テクノ，ファンク，ソウル，ビッグバンド，モダンジャズ，フュージョン，ボサノバ，サンバ，レゲエ，タンゴ，クラシック，ブラスバンド，ブルース，フォーク，カントリー，ゴスペル，アフリカン，インド，フラメンコ，シャンソン，カンツォーネ，演歌，民謡，雅楽，ア・カペラの 31 ジャンルを用いた．

MIDI データにはヘッダ部とトラックデータ部が存在するが，ヘッダ部はトラックデータ部と比べてデータ量が遥かに小さいため，無視できる程度の誤差と考え特別な処理を省略した．また，トラックデータ部には音の高さ，音の長さ，音の強さ，システムコマンドの 4 種類のデータが存在するが，音の高さ，音の長さ，音の強さは 0x00～0x7F の範囲，システムコマンドは 0x80～0xFF の範囲で表されるため，0x00～0x7F の 128 文字のデータを用いることによってシステムコマンドを取り除く．

メロディーとジャンルの間に何かしらの関係性がある場合，それぞれのジャンルによって音の高さと音の長さ，音の強さの特徴があると考えられるが，音の高さ，長さ，強さは独立した情報であるために特徴は異なり，音の高さと音の長さ，音の強さの分布を足し合わせて一つの分布として見てもジャンルごとの特徴が失われないと考えた．よって，MIDI データの音の高さと音の長さ，音の強さの情報を分離せずに用いて分類を行ってもある程度の性能で分類が可能であると考えた．

MIDI データのバイナリデータから得た 16 進数 2 桁の各文字のヒストグラムを用いた．音高，音の長さ，音の強さは 0x00～0x7F の範囲のみで表されるため，0x80～0xFF の 128 文字のデータを除いた 128 次元のベクトルを作成して 1 曲分のデータとした．

5.1.2.実験 1 の結果

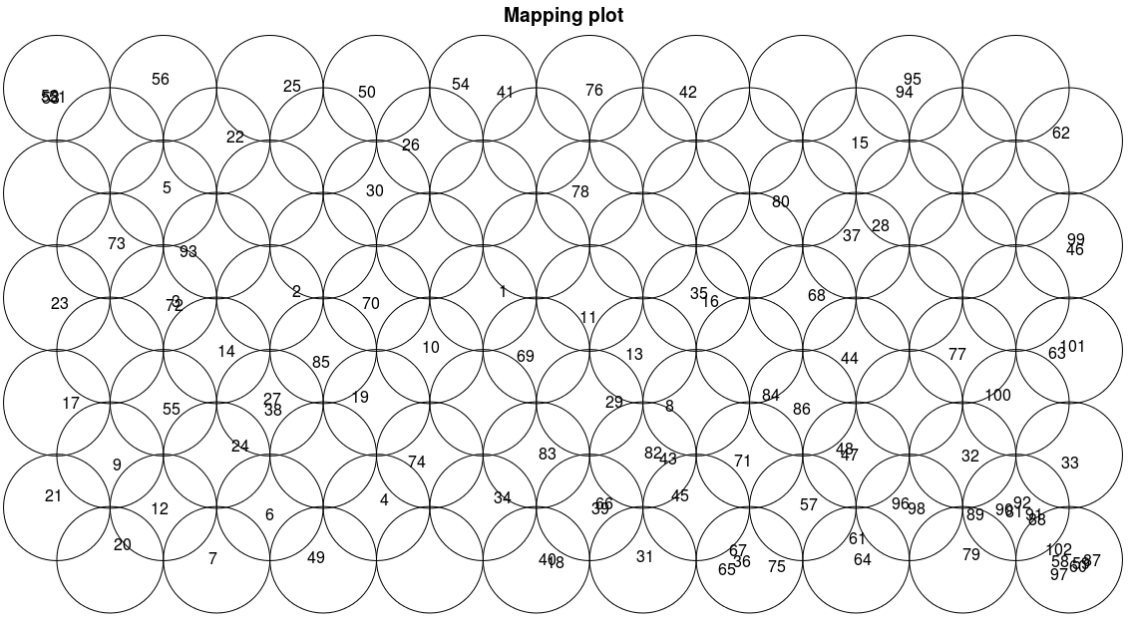

図 5.1：入力 128 次元，出力 2 次元 100 ユニットの出力結果

図 5.1 は実験 1 の SOM である．図より，30 分類中 23 分類，101 楽曲中 61 楽曲(約 60.4%)が同ジャンルの他楽曲と近い位置に配置されたことが確認された．同ジャンルの楽曲と同じユニットに出力されたのは 101 楽曲中 22 楽曲である．なお，ア・カペラというジャンルの楽曲は 1 曲のみであるため，分類・楽曲の総数から除いた．

例えば図の下から 2 段目，1 番右のユニットを見ると 90，91，92 が同じノードにあり，それぞれカンツォーネであるので SOM の分類とデータのジャンルが一致していることがわかる．

表 5.1：楽曲番号とジャンル対応表

楽曲番号	大分類	中分類
No. 1	ポップス	ポップス
No. 2		
No. 3		
No. 4		バラード
No. 5		
No. 6		
No. 7	ロック	ロック
No. 8		
No. 9		
No. 10		ヘビーメタル
No. 11		
No. 12		
No. 13	ダンス	ラップ／ヒップホップ
No. 14		
No. 15		
No. 16		ハウス
No. 17		
No. 18		
No. 19		テクノ
No. 20		
No. 21		
No. 22		ファンク
No. 23		
No. 24		
No. 25		ソウル／R＆B
No. 26		
No. 27		
No. 28	ジャズ	ビッグバンド
No. 29		
No. 30		
No. 31		モダンジャズ
No. 32		
No. 33		
No. 34		フュージョン
No. 35		
No. 36		
No. 37	ラテン	ボサノバ
No. 38		
No. 39		
No. 40		サンバ
No. 41		
No. 42		
No. 43		レゲエ
No. 44		
No. 45		
No. 46		タンゴ
No. 47		
No. 48		
No. 49	クラシック	バロック(管弦楽)
No. 50		古典派(管弦楽)
No. 51		ロマン派(管弦楽)
No. 52		近代(管弦楽)
No. 53		
No. 54	行進曲	ブラスバンド
No. 55		
No. 56		
No. 57	クラシック	バロック（器楽）
No. 58		
No. 59		
No. 60		
No. 61		古典派（器楽）
No. 62		古典派(室内楽)
No. 63		ロマン派(室内楽)
No. 64		ロマン派（器楽）
No. 65		近代(器楽)
No. 66	ワールド	ブルース
No. 67		
No. 68		
No. 69		フォーク
No. 70		
No. 71		

No. 72	ワールド	カントリー
No. 73		
No. 74		
No. 75		ゴスペル
No. 76		
No. 77		
No. 78		アフリカン
No. 79		
No. 80		
No. 81		インド
No. 82		
No. 83		
No. 84		フラメンコ
No. 85		
No. 86		
No. 87	声楽	シャンソン
No. 88		
No. 89		
No. 90		カンツォーネ
No. 91		
No. 92		
No. 93	邦楽	演歌
No. 94		
No. 95		
No. 96		民謡
No. 97		
No. 98		
No. 99		雅楽
No. 100		
No. 101		
No. 102	ア・カペラ	ア・カペラ

表 5.2：SOM 上にて同ジャンル付近に配置された楽曲(128 次元)

楽曲番号	大分類	中分類
No. 1	ポップス	ポップス
No. 2		
No. 3		
No. 4		バラード
No. 6		
No. 7	ロック	ロック
No. 9		
No. 10		ヘビーメタル
No. 11		
No. 20	ダンス	テクノ
No. 21		
No. 25		ソウル／R&B
No. 26		
No. 32	ジャズ	モダンジャズ
No. 33		
No. 40	ラテン	サンバ
No. 41		
No. 42		
No. 43		レゲエ
No. 45		
No. 47		タンゴ
No. 48		
No. 49	クラシック	バロック(管弦楽)
No. 50		古典派(管弦楽)
No. 51		ロマン派(管弦楽)
No. 52		近代(管弦楽)
No. 53		
No. 57		バロック（器楽）
No. 58		
No. 59		
No. 60		
No. 61		古典派（器楽）
No. 62		古典派(室内楽)
No. 63		ロマン派(室内楽)
No. 64		ロマン派（器楽）
No. 65		近代(器楽)
No. 66	ワールド	ブルース
No. 67		
No. 69		フォーク
No. 70		
No. 72		カントリー
No. 73		
No. 75		ゴスペル
No. 76		
No. 78		アフリカン
No. 80		
No. 82		インド
No. 83		
No. 84		フラメンコ
No. 86		
No. 87	声楽	シャンソン
No. 88		
No. 89		
No. 90		カンツォーネ
No. 91		
No. 92		
No. 94	邦楽	演歌
No. 95		
No. 96		民謡
No. 97		
No. 98		
No. 99		雅楽
No. 100		
No. 101		

5.2.実験 2

さらなる計算量削減を考える．音の高さ，長さ，強さはそれぞれ 7bit128 階調の 16 進数 2 桁で表現されるため，実験 1 では 128 次元のベクトルを用いた．しかし，多くの楽曲はその一部分の値のみを用いている事に着目し，桁を無視して MIDI データを 8bit16 進数 1 桁の集合とみなして読み込み, 16 次元のヒストグラムをデータベクトルとしても楽曲の特徴を保持しているのではないかと考えた．これにより，MIDI データからのヒストグラムの作成時の処理を低減することができる．さらに 128 次元データの 8 分の 1 の 16 次元データを用いて SOM の学習を行うため学習時間が短縮される．実験 2 では実験 1 と同じ手順，パラメータで行う全データについて，128 次元ではなく 16 次元に作成しなおして用いた．

5.2.1.実験 2 のデータ

MIDI データのバイナリデータから得た 16 進数 2 桁の各文字のヒストグラムを用いた．16 次元のベクトルを作成して 1 曲分のデータとした．なお，本実験では処理の軽減を優先して全ての文字を用いたため，実験 1 では除いたシステムコマンド 0x80～0xFF の範囲の値も含まれている．

5.2.2.実験 2 の結果

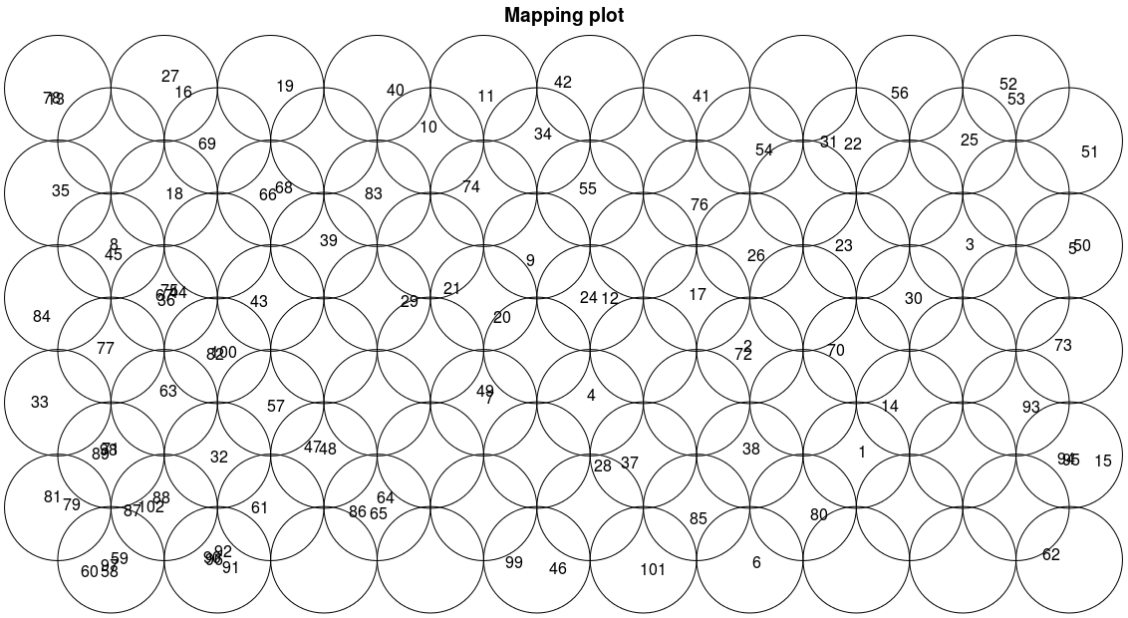

図 5.2：入力 16 次元，出力 2 次元 100 ユニットの出力結果

図 5.2 は実験 2 の SOM である．図より，30 分類中 22 分類，101 楽曲中 63 楽曲(約 62.4%)が同ジャンルの他楽曲と近い位置に配置されたことが確認された．同ジャンルの楽曲と同じユニットに出力されたのは 101 楽曲中 20 楽曲である．なお，ア・カペラというジャンルの楽曲は 1 曲のみであるため，分類・楽曲の総数から除いた．

例えば図の左下のユニットを見ると 90，91，92 が同じノードにあり，それぞれカンツォーネであるので SOM の分類とデータのジャンルが一致していることがわかる．

表 5.3：SOM 上にて同ジャンル付近に配置された楽曲(16 次元)

楽曲番号	大分類	中分類
No. 1	ポップス	ポップス
No. 2		
No. 10	ロック	ヘビーメタル
No. 11		
No. 14	ダンス	ラップ／ヒップホップ
No. 15		
No. 16		ハウス
No. 18		
No. 20		テクノ
No. 21		
No. 22		ファンク
No. 23		
No. 32	ジャズ	モダンジャズ
No. 33		
No. 35		フュージョン
No. 36		
No. 37	ラテン	ボサノバ
No. 38		
No. 40		サンバ
No. 41		
No. 42		
No. 43		レゲエ
No. 44		
No. 45		
No. 47		タンゴ
No. 48		
No. 50	クラシック	古典派(管弦楽)
No. 51		ロマン派(管弦楽)
No. 52		近代(管弦楽)
No. 53		
No. 54	行進曲	ブラスバンド
No. 55		
No. 56		
No. 57	クラシック	バロック（器楽）
No. 58		バロック（器楽）
No. 59		バロック（器楽）
No. 60		バロック（器楽）
No. 61		古典派（器楽）
No. 62		古典派(室内楽)
No. 63		ロマン派(室内楽)
No. 64		ロマン派（器楽）
No. 65		近代(器楽)
No. 66	ワールド	ブルース
No. 67		
No. 75		ゴスペル
No. 77		
No. 78		アフリカン
No. 79		
No. 87	声楽	シャンソン
No. 88		
No. 89		
No. 90		カンツォーネ
No. 91		
No. 92		
No. 93	邦楽	演歌
No. 94		
No. 95		
No. 96		民謡
No. 97		
No. 98		
No. 99		雅楽
No. 101		

6.考察

実験 1 においては約 60.4%，実験 2 においては約 62.4%の楽曲が分類できることが確認された．ヒストグラムを作成する処理時間は 128 次元と比べて 16 次元は約 6.28%減少した．精度という観点では，同じユニットに出力された数の多い 128 次元の方が優れているが，分類数という観点では 16 次元の方が優れていると言える．これは，128 次元から 16 次元へと変化した際に音楽と関係のない 0x80～0xFF のデータが増加したことにより，汎用性が高くなる代わりに精度が減少したと推測される．しかし，精度においても分類数においてもほとんど差のないことを考慮すると 16 次元で扱っても問題はないと考える．処理時間が短い点や本研究での目的においては精度よりも汎用性を重視する点を考慮すると，今回の目的には 16 次元の方が適している．分類システムとして利用可能なレベルではないが，過半数の楽曲が分類可能ということは，MIDI データから得られるメロディーとジャンルとの間に何らかの関係性があることが示唆される．

7.結論

本研究では，SOM と MIDI データを用いた曲調分類を検討した．MIDI データからバイナリデータを取得し，そこから音の高さ，長さ，強さの特徴量を抽出し，その情報を入力として SOM を作成し，曲調ごとに分類を行う手法である．多くのジャンルの曲の音の高さ，長さ，強さの部分を 16 進数 2 桁もしくは 16 進数 1 桁で取り出してヒストグラムを作成，そのヒストグラムを入力として SOM の作成を行った．その実験から 16 進数 2 桁では約 60.4%，16 進数 1 桁では約 62.4%の楽曲が分類できることが確認された．その結果，メロディーとジャンルの間に何らかの関係があることが示唆された．また，16 次元と 128 次元の 2 種類の実験を行うことにより，16 次元の情報量の少ないデータでも充分有用であることを確認した．

今後は曲調に対しての評価実験を行って汎用性を確認すること，実際に MIDI 楽器を演奏した時の演奏データを用いて SOM を作成すること，簡易的な処理のまま分類精度を上げる方法についての検討を行っていきたい．

謝辞

本研究を進めるにあたり，多くのご指導，ご助言を頂いた卒業論文指導教員の三好力教授に厚く御礼申し上げます．また，多くの示唆を頂いた三好研究室の皆様や友人に感謝致します.

参考文献

[1]高橋 信之，コンプリート MIDI ブック，株式会社リットーミュージック，2005
[2]遠藤 司，伊藤 伸一，満倉 靖恵，福見 稔，“曲調解析に基づく音楽分析法の提案”，電子情報通信学会技術研究報告．NC，ニューロコンピューティング 108(101)，51-54(2008)

付録

端末上のコマンド

```
hexdump -C /*バイナリデータの取得*/
sed -e “s/^.....//” /*ファイルの行頭から 8 字削除*/
sed -e “s/.....¥$//” /*ファイルの行末から 20 字削除*/
sed -e ‘$d’ /*ファイルの末尾を 1 行削除*/
```

128 次元のカウントプログラム

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
   FILE *fp, *fp2;
            char *fname = "rm-g001no.txt"; /* 001 を対象のファイル名に適宜変更  */
   int c, x=0, j;
   int n0='0', nc00=0, n1='1', nc01=0, n2='2', nc02=0, n3='3', nc03=0, n4='4', nc04=0, n5='5', nc05=0;
   int n6='6', nc06=0, n7='7', nc07=0, n8='8', nc08=0, n9='9', nc09=0, na='a', nc0a=0, nb='b', nc0b=0;
   int nc='c', nc0c=0, nd='d', nc0d=0, ne='e', nc0e=0, nf='f', nc0f=0, nc10=0, nc11=0, nc12=0, nc13=0;
   int nc14=0, nc15=0, nc16=0, nc17=0, nc18=0, nc19=0, nc1a=0, nc1b=0, nc1c=0, nc1d=0, nc1e=0, nc1f=0;
   int nc20=0, nc21=0, nc22=0, nc23=0, nc24=0, nc25=0, nc26=0, nc27=0, nc28=0, nc29=0, nc2a=0, nc2b=0;
   int nc2c=0, nc2d=0, nc2e=0, nc2f=0, nc30=0, nc31=0, nc32=0, nc33=0, nc34=0, nc35=0, nc36=0, nc37=0;
   int nc38=0, nc39=0, nc3a=0, nc3b=0, nc3c=0, nc3d=0, nc3e=0, nc3f=0, nc40=0, nc41=0, nc42=0, nc43=0;
   int nc44=0, nc45=0, nc46=0, nc47=0, nc48=0, nc49=0, nc4a=0, nc4b=0, nc4c=0, nc4d=0, nc4e=0, nc4f=0;
   int nc50=0, nc51=0, nc52=0, nc53=0, nc54=0, nc55=0, nc56=0, nc57=0, nc58=0, nc59=0, nc5a=0, nc5b=0;
   int nc5c=0, nc5d=0, nc5e=0, nc5f=0, nc60=0, nc61=0, nc62=0, nc63=0, nc64=0, nc65=0, nc66=0, nc67=0;
   int nc68=0, nc69=0, nc6a=0, nc6b=0, nc6c=0, nc6d=0, nc6e=0, nc6f=0, nc70=0, nc71=0, nc72=0, nc73=0;
   int nc74=0, nc75=0, nc76=0, nc77=0, nc78=0, nc79=0, nc7a=0, nc7b=0, nc7c=0, nc7d=0, nc7e=0, nc7f=0;
   int nc80=0, nc81=0, nc82=0, nc83=0, nc84=0, nc85=0, nc86=0, nc87=0, nc88=0, nc89=0, nc8a=0, nc8b=0;
   int nc8c=0, nc8d=0, nc8e=0, nc8f=0, nc90=0, nc91=0, nc92=0, nc93=0, nc94=0, nc95=0, nc96=0, nc97=0;
   int nc98=0, nc99=0, nc9a=0, nc9b=0, nc9c=0, nc9d=0, nc9e=0, nc9f=0, nca0=0, nca1=0, nca2=0, nca3=0;
   int nca4=0, nca5=0, nca6=0, nca7=0, nca8=0, nca9=0, ncaa=0, ncab=0, ncac=0, ncad=0, ncae=0, ncaf=0;
   int ncb0=0, ncb1=0, ncb2=0, ncb3=0, ncb4=0, ncb5=0, ncb6=0, ncb7=0, ncb8=0, ncb9=0, ncba=0, ncbb=0;
   int ncbc=0, ncbd=0, ncbe=0, ncbf=0, ncc0=0, ncc1=0, ncc2=0, ncc3=0, ncc4=0, ncc5=0, ncc6=0, ncc7=0;
   int ncc8=0, ncc9=0, ncca=0, nccb=0, nccc=0, nccd=0, ncce=0, nccf=0, ncd0=0, ncd1=0, ncd2=0, ncd3=0;
   int ncd4=0, ncd5=0, ncd6=0, ncd7=0, ncd8=0, ncd9=0, ncda=0, ncdb=0, ncdc=0, ncdd=0, ncde=0, ncdf=0;
   int nce0=0, nce1=0, nce2=0, nce3=0, nce4=0, nce5=0, nce6=0, nce7=0, nce8=0, nce9=0, ncea=0, nceb=0, ncec=0;
   int nced=0, ncee=0, ncef=0, ncf0=0, ncf1=0, ncf2=0, ncf3=0, ncf4=0, ncf5=0, ncf6=0, ncf7=0, ncf8=0;
   int ncf9=0, ncfa=0, ncfb=0, ncfc=0, ncfd=0, ncfe=0, ncff=0;
   int flg='x';

            if((fp = fopen(fname, "r")) == NULL) {
                  printf("Open error!¥n");
                  return(-1);
            }
   if((fp2 = fopen("128num.txt", "a+")) == NULL) {
      printf("128num Open error!¥n");
      return(-1);
   }

   while((c = fgetc(fp)) != EOF) {
      if(c == n0) {
         if(flg == 'x') flg = n0;
         else {
            if(flg == n0) {
               nc00++;
               flg = 'x';
            }
            else if(flg == n1) {
               nc10++;
               flg = 'x';
            }
            else if(flg == n2) {
               nc20++;
               flg = 'x';
            }
            else if(flg == n3) {
               nc30++;
               flg = 'x';
            }
            else if(flg == n4) {
               nc40++;
               flg = 'x';
            }
            else if(flg == n5) {
               nc50++;
               flg = 'x';
            }
            else if(flg == n6) {
               nc60++;
               flg = 'x';
            }
            else if(flg == n7) {
               nc70++;
               flg = 'x';
            }
            else flg = 'x';
      }
   }
   else if(c == n1) {
      if(flg == 'x') flg = n1;
      else {
            if(flg == n0) {
               nc01++;
               flg = 'x';
            }
            else if(flg == n1) {
               nc11++;
               flg = 'x';
            }
            else if(flg == n2) {
               nc21++;
               flg = 'x';
            }
            else if(flg == n3) {
               nc31++;
               flg = 'x';
            }
            else if(flg == n4) {
               nc41++;
               flg = 'x';
            }
            else if(flg == n5) {
               nc51++;
               flg = 'x';
            }
            else if(flg == n6) {
               nc61++;
               flg = 'x';
            }
            else if(flg == n7) {
               nc71++;
               flg = 'x';
            }
            else flg = 'x';
      }
            }
   else if(c == n2) {
      if(flg == 'x') flg = n2;
      else {
            if(flg == n0) {
               nc02++;
               flg = 'x';
            }
            else if(flg == n1) {
               nc12++;
               flg = 'x';
            }
            else if(flg == n2) {
               nc22++;
               flg = 'x';
```

```
            }
            else if(flg == n3) {
               nc32++;
               flg = 'x';
            }
            else if(flg == n4) {
               nc42++;
               flg = 'x';
            }
            else if(flg == n5) {
               nc52++;
               flg = 'x';
            }
            else if(flg == n6) {
               nc62++;
               flg = 'x';
            }
            else if(flg == n7) {
               nc72++;
               flg = 'x';
            }
            else flg = 'x';
      }
   }
   else if(c == n3) {
      if(flg == 'x') flg = n3;
      else {
            if(flg == n0) {
               nc03++;
               flg = 'x';
            }
            else if(flg == n1) {
               nc13++;
               flg = 'x';
            }
            else if(flg == n2) {
               nc23++;
               flg = 'x';
            }
            else if(flg == n3) {
               nc33++;
               flg = 'x';
            }
            else if(flg == n4) {
               nc43++;
               flg = 'x';
            }
            else if(flg == n5) {
               nc53++;
               flg = 'x';
            }
            else if(flg == n6) {
               nc63++;
               flg = 'x';
            }
            else if(flg == n7) {
               nc73++;
               flg = 'x';
            }
            else flg = 'x';
      }
   }
   else if(c == n4) {
      if(flg == 'x') flg = n4;
      else {
            if(flg == n0) {
               nc04++;
               flg = 'x';
            }
            else if(flg == n1) {
               nc14++;
               flg = 'x';
            }
            else if(flg == n2) {
               nc24++;
               flg = 'x';
            }
            else if(flg == n3) {
               nc34++;
               flg = 'x';
            }
            else if(flg == n4) {
               nc44++;
               flg = 'x';
            }
            else if(flg == n5) {
               nc54++;
               flg = 'x';
            }
            else if(flg == n6) {
               nc64++;
               flg = 'x';
            }
            else if(flg == n7) {
               nc74++;
               flg = 'x';
            }
            else flg = 'x';
      }
   }
   else if(c == n5) {
      if(flg == 'x') flg = n5;
      else {
            if(flg == n0) {
               nc05++;
               flg = 'x';
            }
            else if(flg == n1) {
               nc15++;
               flg = 'x';
            }
            else if(flg == n2) {
               nc25++;
               flg = 'x';
            }
            else if(flg == n3) {
               nc35++;
               flg = 'x';
            }
            else if(flg == n4) {
               nc45++;
               flg = 'x';
            }
            else if(flg == n5) {
               nc55++;
               flg = 'x';
            }
            else if(flg == n6) {
               nc65++;
               flg = 'x';
            }
            else if(flg == n7) {
               nc75++;
               flg = 'x';
            }
            else flg = 'x';
      }
   }
   else if(c == n6) {
      if(flg == 'x') flg = n6;
      else {
            if(flg == n0) {
               nc06++;
               flg = 'x';
            }
            else if(flg == n1) {
               nc16++;
               flg = 'x';
            }
            else if(flg == n2) {
               nc26++;
               flg = 'x';
            }
            else if(flg == n3) {
               nc36++;
               flg = 'x';
            }
            else if(flg == n4) {
               nc46++;
               flg = 'x';
            }
            else if(flg == n5) {
               nc56++;
               flg = 'x';
            }
            else if(flg == n6) {
               nc66++;
               flg = 'x';
            }
            else if(flg == n7) {
```

```
                nc76++;
                flg = 'x';
            }
            else flg = 'x';
    }
}
else if(c == n7) {
    if(flg == 'x') flg = n7;
    else {
            if(flg == n0) {
                nc07++;
                flg = 'x';
            }
            else if(flg == n1) {
                nc17++;
                flg = 'x';
            }
            else if(flg == n2) {
                nc27++;
                flg = 'x';
            }
            else if(flg == n3) {
                nc37++;
                flg = 'x';
            }
            else if(flg == n4) {
                nc47++;
                flg = 'x';
            }
            else if(flg == n5) {
                nc57++;
                flg = 'x';
            }
            else if(flg == n6) {
                nc67++;
                flg = 'x';
            }
            else if(flg == n7) {
                nc77++;
                flg = 'x';
            }
            else flg = 'x';
    }
}
else if(c == n8) {
    if(flg == 'x') flg = n8;
    else {
            if(flg == n0) {
                nc08++;
                flg = 'x';
            }
            else if(flg == n1) {
                nc18++;
                flg = 'x';
            }
            else if(flg == n2) {
                nc28++;
                flg = 'x';
            }
            else if(flg == n3) {
                nc38++;
                flg = 'x';
            }
            else if(flg == n4) {
                nc48++;
                flg = 'x';
            }
            else if(flg == n5) {
                nc58++;
                flg = 'x';
            }
            else if(flg == n6) {
                nc68++;
                flg = 'x';
            }
            else if(flg == n7) {
                nc78++;
                flg = 'x';
            }
            else flg = 'x';
    }
}
else if(c == n9) {
    if(flg == 'x') flg = n9;
    else {
            if(flg == n0) {
                nc09++;
                flg = 'x';
            }
            else if(flg == n1) {
                nc19++;
                flg = 'x';
            }
            else if(flg == n2) {
                nc29++;
                flg = 'x';
            }
            else if(flg == n3) {
                nc39++;
                flg = 'x';
            }
            else if(flg == n4) {
                nc49++;
                flg = 'x';
            }
            else if(flg == n5) {
                nc59++;
                flg = 'x';
            }
            else if(flg == n6) {
                nc69++;
                flg = 'x';
            }
            else if(flg == n7) {
                nc79++;
                flg = 'x';
            }
            else flg = 'x';
    }
}
else if(c == na) {
    if(flg == 'x') flg = na;
    else {
            if(flg == n0) {
                nc0a++;
                flg = 'x';
            }
            else if(flg == n1) {
                nc1a++;
                flg = 'x';
            }
            else if(flg == n2) {
                nc2a++;
                flg = 'x';
            }
            else if(flg == n3) {
                nc3a++;
                flg = 'x';
            }
            else if(flg == n4) {
                nc4a++;
                flg = 'x';
            }
            else if(flg == n5) {
                nc5a++;
                flg = 'x';
            }
            else if(flg == n6) {
                nc6a++;
                flg = 'x';
            }
            else if(flg == n7) {
                nc7a++;
                flg = 'x';
            }
            else flg = 'x';
    }
}
else if(c == nb) {
    if(flg == 'x') flg = nb;
    else {
            if(flg == n0) {
                nc0b++;
                flg = 'x';
            }
            else if(flg == n1) {
                nc1b++;
                flg = 'x';
            }
            else if(flg == n2) {
```

```
                nc2b++;
                flg = 'x';
            }
            else if(flg == n3) {
                nc3b++;
                flg = 'x';
            }
            else if(flg == n4) {
                nc4b++;
                flg = 'x';
            }
            else if(flg == n5) {
                nc5b++;
                flg = 'x';
            }
            else if(flg == n6) {
                nc6b++;
                flg = 'x';
            }
            else if(flg == n7) {
                nc7b++;
                flg = 'x';
            }
            else flg = 'x';
    }
}
else if(c == nc) {
    if(flg == 'x') flg = nc;
    else {
            if(flg == n0) {
                nc0c++;
                flg = 'x';
            }
            else if(flg == n1) {
                nc1c++;
                flg = 'x';
            }
            else if(flg == n2) {
                nc2c++;
                flg = 'x';
            }
            else if(flg == n3) {
                nc3c++;
                flg = 'x';
            }
            else if(flg == n4) {
                nc4c++;
                flg = 'x';
            }
            else if(flg == n5) {
                nc5c++;
                flg = 'x';
            }
            else if(flg == n6) {
                nc6c++;
                flg = 'x';
            }
            else if(flg == n7) {
                nc7c++;
                flg = 'x';
            }
            else flg = 'x';
    }
}
else if(c == nd) {
    if(flg == 'x') flg = nd;
    else {
            if(flg == n0) {
                nc0d++;
                flg = 'x';
            }
            else if(flg == n1) {
                nc1d++;
                flg = 'x';
            }
            else if(flg == n2) {
                nc2d++;
                flg = 'x';
            }
            else if(flg == n3) {
                nc3d++;
                flg = 'x';
            }
            else if(flg == n4) {
                nc4d++;
                flg = 'x';
            }
            else if(flg == n5) {
                nc5d++;
                flg = 'x';
            }
            else if(flg == n6) {
                nc6d++;
                flg = 'x';
            }
            else if(flg == n7) {
                nc7d++;
                flg = 'x';
            }
            else flg = 'x';
    }
}
else if(c == ne) {
    if(flg == 'x') flg = ne;
    else {
            if(flg == n0) {
                nc0e++;
                flg = 'x';
            }
            else if(flg == n1) {
                nc1e++;
                flg = 'x';
            }
            else if(flg == n2) {
                nc2e++;
                flg = 'x';
            }
            else if(flg == n3) {
                nc3e++;
                flg = 'x';
            }
            else if(flg == n4) {
                nc4e++;
                flg = 'x';
            }
            else if(flg == n5) {
                nc5e++;
                flg = 'x';
            }
            else if(flg == n6) {
                nc6e++;
                flg = 'x';
            }
            else if(flg == n7) {
                nc7e++;
                flg = 'x';
            }
            else flg = 'x';
    }
}
else if(c == nf) {
    if(flg == 'x') flg = nf;
    else {
            if(flg == n0) {
                nc0f++;
                flg = 'x';
            }
            else if(flg == n1) {
                nc1f++;
                flg = 'x';
            }
            else if(flg == n2) {
                nc2f++;
                flg = 'x';
            }
            else if(flg == n3) {
                nc3f++;
                flg = 'x';
            }
            else if(flg == n4) {
                nc4f++;
                flg = 'x';
            }
            else if(flg == n5) {
                nc5f++;
                flg = 'x';
            }
            else if(flg == n6) {
                nc6f++;
                flg = 'x';
```

```
                }
                else if(flg == n7) {
                   nc7f++;
                   flg = 'x';
                }
                else flg = 'x';
          }
       }
    }

   fprintf(fp2, "%d %d %d ",nc00,nc01,nc02);
   fprintf(fp2,    "%d    %d    %d    %d    %d    %d    %d
",nc03,nc04,nc05,nc06,nc07,nc08,nc09);
   fprintf(fp2,    "%d    %d    %d    %d    %d    %d
",nc0a,nc0b,nc0c,nc0d,nc0e,nc0f);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc10,nc11,nc12,nc13,nc14,nc15,nc16,nc17);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc18,nc19,nc1a,nc1b,nc1c,nc1d,nc1e,nc1f);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc20,nc21,nc22,nc23,nc24,nc25,nc26,nc27);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc28,nc29,nc2a,nc2b,nc2c,nc2d,nc2e,nc2f);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc30,nc31,nc32,nc33,nc34,nc35,nc36,nc37);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc38,nc39,nc3a,nc3b,nc3c,nc3d,nc3e,nc3f);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc40,nc41,nc42,nc43,nc44,nc45,nc46,nc47);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc48,nc49,nc4a,nc4b,nc4c,nc4d,nc4e,nc4f);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc50,nc51,nc52,nc53,nc54,nc55,nc56,nc57);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc58,nc59,nc5a,nc5b,nc5c,nc5d,nc5e,nc5f);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc60,nc61,nc62,nc63,nc64,nc65,nc66,nc67);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc68,nc69,nc6a,nc6b,nc6c,nc6d,nc6e,nc6f);
   fprintf(fp2,  "%d  %d  %d  %d  %d  %d  %d  %d
",nc70,nc71,nc72,nc73,nc74,nc75,nc76,nc77);
   fprintf(fp2,
"%d %d %d %d %d %d %d %d¥n",nc78,nc79,nc7a,nc7b,nc7c,n
c7d,nc7e,nc7f);
   }

   fclose(fp);
   fclose(fp2);
   return(0);
}
```

16 次元のカウントプログラム

```
#include <stdio.h>

int main(void) {
                FILE *fp;
                char *fname = "rm-g100no.txt";
                int c, x=0;
               int n0='0', nc00=0, n1='1', nc01=0, n2='2', nc02=0,
n3='3', nc03=0, n4='4', nc04=0, n5='5', nc05=0;
               int n6='6', nc06=0, n7='7', nc07=0, n8='8', nc08=0,
n9='9', nc09=0, na='a', nc0a=0, nb='b', nc0b=0;
                int nc='c', nc0c=0, nd='d', nc0d=0, ne='e', nc0e=0,
nf='f', nc0f=0;

                fp = fopen(fname, "r");
                if(fp == NULL) {
                        printf("Open error!¥n");
                        return(-1);
                }

                while((c = fgetc(fp)) != EOF) {
                   if(c == n0) nc00++;
                   if(c == n1) nc01++;
                   if(c == n2) nc02++;
                   if(c == n3) nc03++;
                   if(c == n4) nc04++;
                   if(c == n5) nc05++;
                   if(c == n6) nc06++;
                   if(c == n7) nc07++;
                   if(c == n8) nc08++;
                   if(c == n9) nc09++;
                   if(c == na) nc0a++;
                   if(c == nb) nc0b++;
                   if(c == nc) nc0c++;
                   if(c == nd) nc0d++;
                   if(c == ne) nc0e++;
                   if(c == nf) nc0f++;
                }

printf("%d %d %d %d %d %d %d %d",nc00,nc01,nc02,nc03,nc
04,nc05,nc06,nc07);

printf(" %d %d %d %d %d %d %d %d¥n",nc08,nc09,nc0a,nc0b
,nc0c,nc0d,nc0e,nc0f);

                fclose(fp);
                return(0);
}
```

R 上のコマンド

```
library(kohonen)
library(gplot2)
x <- read.table(“/128.txt”, header=F)
UseData <-do.call(cbind, x)
GRID <- somgrid(xdim = 10, ydim = 10, topo = “hexa”)
DataSom <- som(UseData, grid = GRID)
plot(DataSom, type = “mapping”, labels = 1:nrow(UseData))
```