最大確率分割情報量を用いた楽曲のクラス分類と時系列データ分類への応用

(Music Classification using Maximum Probability Segmentation and its Application to Time Series Data)

2024 年 1 月

博士（工学）

高本 綺架

豊橋技術科学大学

別紙４－２（課程博士（和文））

2024 年 1 月 4 日

情報・知能工学専攻	学籍番号	第 219302 号	指導教員	梅村　恭司 北岡　教英
氏名	高本　綺架			

論文内容の要旨 （博士）

博士学位論文名	最大確率分割情報量を用いた楽曲のクラス分類と時系列データ分類への応用

（要旨　1,200 字程度）

人間は長い歴史の中で様々な創造物を生み出してきた．これらの創造物は，音楽や文学作品など多岐に渡る．多くの作品には，作った人間の個性や思考が反映されている．この創造物に含まれる作者の固有の表現を見つけることは，興味深い問題である．本研究では，この人間の創造物に含まれる特徴に焦点をあて，それらを計算機科学の観点から探究することを目的とする．

一般に，分類は未知のデータが属するクラスを，既知のデータを用いて推定する．その際，使用されている分類手法は，データの中から何かしらの特徴を捉え．それをもとにデータを分類していると考えられる．つまり，データの分類が可能であるということは，データに含まれる特徴を捉えていることになる．データの分類手法として，様々な手法が提案されているが，我々は文字列処理の手法を応用するアプローチを提案する．

提案した手法が実際に分類尺度として機能するかを音楽データと時系列データを対象として調査を行う．音楽には様々な種類があるが，特にクラシック音楽に着目する．音楽は，言語に依存しない特徴を持つ創作物の一つであり，歌詞を持たないクラシック音楽は，純粋な音のパターンと構造に作曲者の個性が現れると考えた．そこで本研究では，クラシック音楽に含まれる作曲者の特徴分析を試みる．時系列データでは，公開されている様々な種類のデータセットを対象に，データの特徴とそのデータを生成しているモデルの特徴を調査することを試みる．

音楽データにおける作曲者分類と時系列データ分類において，いずれも提案手法が圧縮を利用した尺度の性能を上回ることを示した．また，作曲者分類においては，作曲者の持つ特徴を頻度情報から実際に分析し，特徴となる長さについて示した．時系列データ分類においては，実際の応用を考慮し，分類性能だけでなく計算速度の観点から高速化を行い，計算量の削減を行った．上記の結果から，全部分文字列の出現頻度をもとに文字列データの情報量を計算する提案手法は，データの分類および特徴分析が可能であることを明らかにした．

本論文は，作曲者分類と時系列データ分類を対象に，最大確率分割情報量の計算を用いた分類手法の有効性を検証した．また，最大確率分割を行う過程で得られた分割方法から，作曲者の特徴的なパターンの長さについても分析を行ったものである．

別紙４－１（課程博士（英文））

Date of Submission（month day, year）： 01 04, 2024

Department of Computer Science and Engineering	Student ID Number	D219302
Applicant's name	Ayaka TAKAMOTO	

Supervisors	Kyoji Umemura Norihide Kitaoka

Abstract（Doctor）

Title of Thesis	Music Classification using Maximum Probability Segmentation and its Application to Time Series Data

Approx. 800 words

Human beings have created a variety of creations throughout their long history. These creations span across different forms such as music and literary works. Many works reflect the personality and thoughts of the creators. Finding the unique expression of the creator in these creations is an interesting problem. This research aims to focus on the features inherent in human creations and explore them from the perspective of computer science.

In general, classification involves estimating the class to which unknown data belongs using known data. In this process, the classification method being used is believed to capture some features from the data and classify it based on them. In other words, the ability to classify data implies capturing the features present in the data. While various methods have been proposed for data classification, we propose an approach that applies string processing techniques.

To investigate whether the proposed method functions as an actual classification scale, we conduct research focusing on music data and time-series data. Although there are various types of music, we particularly emphasize classical music. Music, being a creation with features independent of language, is considered to reveal the composer's personality in the pure patterns and structures of non-lyrical classical music. Therefore, in this study, we attempt to analyze the characteristics of composers in classical music. For time-series data, we aim to investigate the features of the data and the model generating the data across various publicly available datasets.

In both composer classification in music data and time-series data classification, the proposed method demonstrated superior performance to scales utilizing compression. Furthermore, in composer classification, we analyzed the features possessed by composers based on frequency information and presented information regarding the characteristic lengths. In time-series data classification, considering practical applications, we not only improved classification performance but also accelerated the process from the perspective of computational speed. From the results above, the proposed method, which calculates the information content of string data based on the frequency of partial substrings, clarified its capability for data classification and feature analysis.

目次

図目次

表目次

第 1 章

序論：本論文の枠組み

人間は長い歴史の中で様々な創造物を生み出してきた．これらの創造物は，音楽や文学作品など多岐に渡る．多くの作品には，作った人間の個性や思考が反映されている．この創造物に含まれる作者の固有の表現を見つけることは，興味深い問題である．本研究では，この人間の創造物に含まれる特徴に焦点をあて，それらを計算機科学の観点から探究することを目的とする．

人間の創造物に含まれる特徴は，データを生成するモデルの特徴や構造であると言い換えることができる．本研究では，このデータを生成するモデルの例として，音楽と時系列データを用いる．これらのデータの特徴を得るため，それぞれのデータについての分類に焦点を当てる．一般に，分類は未知のデータが属するであろうクラスを，既知のデータを用いて推定する．その際，使用されている分類手法は，データの中から何かしらの特徴を捉え．それをもとにデータを分類していると考えられる．つまり，データの分類が可能であるということは，データに含まれる特徴を捉えていることになる．

データの分類手法として, 様々な手法が提案されているが, 我々は文字列処理の手法を応用するアプローチを用いる．文字列処理の手法についても様々なものがあるが，本研究では圧縮を用いたアルゴリズムである，Compression-based Dissimilarity Measure (CDM) を改良した手法を提案する．CDM は，2 つのデータが類似していれば，個別に 2 つのデータを圧縮するよりも，1 つに連結して圧縮した方がより圧縮できるという，圧縮の原理に基づいている．提案手法は，対象データに含まれる全部分文字列の出現確率を用いて計算することで，データが持つ情報量を計算する．1 対 1 で比較を行う CDM に対し，提案手法はクラスごとに算出された情報量の大小による比較を行うことで，1 対多によるクラス分類を可能にしている．

文字列に含まれる情報量は，判定対象となる文字列に含まれる部分文字列の出現頻度から計算される．この出現確率は，学習データをクラスごとにまとめたデータを母空間として計算される．対象データに含まれるすべての部分文字列の出現頻度を用いて情報量を計算し，その情報量が最小になる，つまり確率が最大になるような組み合わせを求める．これは，対象データのもっとも理想的な分割を求めることに相当する．

圧縮プログラムは，通常自身のデータを利用して圧縮を行うが，本手法では任意の文字列の統計情報を用いることができる．CDM などの類似尺度を用いたクラス分類では，テストデータと近い値のデータからクラスを推定する，k 近傍法などが用いられる．しかし，k 近傍法を用いたクラス分類は，近傍となるデータの選び方に依存するため，近傍としてどの程度の値を選ぶかがバイアスとなっている．一方で，文字列の情報量を計算する本手法は，母空間となる文字列が同一クラスに属するデータ全体との類似度計算に相当するため，このバイアスを軽減しており，CDM によるクラス分類の改良となっている．

提案した手法が実際に分類尺度として機能するかを音楽データと時系列データを対象として調査を行う．音楽には様々な種類があるが，特にクラシック音楽に着目する．音楽は，言語に依存しない特徴を持つ創作物の一つであり，歌詞を持たないクラシック音楽は，純粋な音のパターンと構造に作曲者の個性が現れると考えた．そこで本研究では，クラシック音楽に含まれる作曲者の特徴分析を試みる．また，もう一つの分類対象として，時系列データ分類に着目した．時系列データでは，公開されている様々な種類のデータセットを対象に，データの特徴とそのデータを生成しているモデルの特徴を調査することを試みる．使用した時系列データセットは，様々な動きの軌跡や波形などから構成されており，これらのデータは生成元となったモデルの特徴が現れていると考えられるため，音楽データと同様に対象タスクとして適切であると考えた．

音楽に対して，計算機科学の観点から処理を行う分野として，音楽情報処理がある．音楽情報処理は，音楽の生成，分析，変換および伝達に関する情報を取り扱う多様な技術領域であり，音楽の分類はこの分野においても古典的なタスクである．その分類先は，作曲者だけではなく，時代やジャンル，ユーザの好みなど様々なものがある．作曲者の分類は，分類先がその楽曲を作成した作曲者であるという決まったものであるにもかかわらず，専門家であっても難しいタスクである．

提案手法を用いて，実際にクラシック音楽を対象に楽曲の分類と，特徴的な文字列の分析が可能であるかを調査する必要がある．本研究では，古典派の作曲者の中でも作風が類似しており，音楽分野の専門家であっても分類が特に難しい作曲者として，ハイドンとモーツァルトを対象に分類を行い，他の先行研究との比較を行った．また，それぞれの作曲者が持つ特徴的なフレーズを抽出し，特徴的なフレーズの長さの観点から分析を行った．この結果から，最大確率分割情報量の計算を用いた分類は，楽曲の中に出現する作曲者固有のパターンやモチーフを捉えていると考えられる．

情報量計算がデータ中に出現するモチーフを捉えることができているならば，同様の特徴を持つ時系列データにも応用が可能であると考えられる．そこで，一般的なデータとして公開されている時系列データを対象に分類を行った．使用するデータとして，UCR TimeSeries [Dau 18] に掲載されている 128 種類の時系列データを対象として，分類を行った．比較手法としては，圧縮を用いた分類尺度として代表的なものであり，同一デー

タセットに対して分類実績のある CDM を用いた．128 種類の時系列データを対象に，最大確率分割情報量と CDM を用いて分類を行い，結果を比較した．実験の結果，最大確率分割情報量を用いた分類が CDM と比較して，統計的にも有意に性能が向上したことがわかった．

最大確率分割情報量は文字列に出現するあらゆる分割における部分文字列の出現頻度を計算する．対象となる文字列を T_1，母空間に相当する文字列を T_2 とした場合，T_1 を分割する方法は $2^{|T_1|-1}$ 通り存在するため，全体で $O(|T_1|^3 \log_2 |T_2|)$ の時間がかかる．実際の時系列データに対して分類を行う場合，その計算速度は重要な課題である．そこで，最大確率分割情報量計算の課題であった計算速度に対して，Suffix Array と動的計画法を用いて計算を工夫することで，計算量を $O(|T_1|^2 \log_2 |T_2|)$ に抑え，高速化を実現した．

最後に，本論文の構成を以下に示す．2 章では，クラス分類や楽曲分類についてのの諸概念について記述する．3 章では，最大確率分割情報量の計算手法の前進となった Compression-based Dissimilarity Measure (CDM) と CDM を用いたクラス分類の問題点及びその対応について述べる．4 章では，提案手法である最大確率分割情報量について，その定義を述べる．5 章では，最大確率分割情報量を用いた楽曲分類について述べ，6 章では，時系列データ分類への応用について述べる．また，7 章で結論を述べる．

第 2 章

前提知識

2.1　対象とする分類タスク

機械学習における分類は，与えられたデータを異なるクラスまたはカテゴリに分類する基本的なタスクであり，機械学習の中でも広く利用されている．分類は，データのラベルが既知であるデータセットを学習に使用し，未知のデータに対して，そのデータがどのクラスやラベルに属するかを予測するものである．具体的には，アルゴリズムやモデルが入力データを解析し，どのクラスに属するかどうかを判定する．本研究では，この分類タスクに焦点を当て，具体的な応用として楽曲の作曲者分類と時系列データ分類の 2 つを取り上げる．

楽曲の分類は，音楽データに対して様々な処理を行う音楽情報処理分野においても古典的なタスクである．楽曲がもつ様々な音響的，構造的な特徴を抽出し，その特徴をもとに複数の楽曲を異なる楽曲を特定のカテゴリやジャンルに分類する．こうした分類は，音楽推薦システムや音楽データベースの検索効率の向上など様々な応用への利用が期待できる重要なタスクである．楽曲の分類は，ユーザの嗜好に合わせた分類や，ジャンル分類など様々な種類があるが，本研究はその中でも作曲者の分類に着目した．作曲者の分類は，その分類先が作曲を行なった人物という決まったラベルであるにも関わらず，その分類は専門家であっても難しいとされている．この問題に対して，様々な手法が提案されている [Herremans 16, Pollastri 01, Micchi 18].

時系列データ分類は，センサーやサウンドなどの様々なアプリケーションによって生成される時系列データを対象としたものである．具体的には，人の動きやネットワークの通信などの時間の経過に伴って変動するものである．これらのデータは，そのカテゴリごとに固有のパターンや特徴を持っており，対象に適した分類アルゴリズムの選択やパラメータ調整が必要である．またこれらのデータは，そのカテゴリごとに固有のパターンや特徴を持っている．この時系列データに含まれるパターンに関する情報が明らかであることは稀であり，対象に適した分類アルゴリズムの選択やパラメータの選択にデータについての事前知識が必要となる．こうした時系列データ分類では，対象となるデータの事前知識を

必要としない手法が望ましい.

2.2　Compression-based Dissimilarity Measure (CDM)

Compression-based Dissimilarity Measure (CDM) は Keogh らの提案するパラメータを必要としないデータマイニングアルゴリズムの一つであり，2 つの文字列間の距離を計算する尺度である．CDM は，式 (2.1) で定義される [Keogh 04b].

$$\mathrm{CDM}(x, y) = \frac{\mathrm{Comp}(xy)}{\mathrm{Comp}(x) + \mathrm{Comp}(y)} \tag{2.1}$$

ここで，x，y は比較対象となる文字列であり，xy は 2 つの文字列を連結したものである．また，$\mathrm{Comp}(x)$，$\mathrm{Comp}(y)$，$\mathrm{Comp}(xy)$ はそれぞれの文字列を圧縮した際のファイルサイズである．文字列 x と文字列 y が類似しているならば CDM の値は小さくなり，逆に 2 つの文字列に関連がなければ CDM の値は 1 に近付く.

CDM は 2 つの文字列間の類似度を図る尺度であり，2 つの文字列 x と y を構成している部分文字列が共通しているほど，文字列 x と y を個別に圧縮し合計した場合より，連結した後に圧縮した場合の方がファイルサイズが小さくなる原理を利用している．圧縮後のファイルサイズは，利用する圧縮プログラムによって変化するが，どのような圧縮プログラムでもランダムな文字列では圧縮率が低く，繰り返しの多い文字列は圧縮率が高くなる．圧縮率が高いほど，圧縮後のファイルサイズは小さくなるため，圧縮後のファイルサイズは，入力された文字列が持つ情報量を反映したものになると考えられる.

多くの圧縮プログラムは，1 文字の頻度を分析するだけでなく，ある長さの文字列の出現についても分析を行っている．そのため，圧縮後のファイルサイズは通信路の理論で扱われるような，文字を符号単位として固定した情報量ではなく，複数の文字からなる単語のような文字列の塊から適切な符号単位を選び出した状態での情報量となる．したがって，圧縮プログラムを使用している CDM は，対象となる文字列の部分文字列を網羅的に分析し類似度を計算する方法を効率よく実装したものである.

2.2.1　CDM によるクラス分類

CDM が利用可能なタスクにはクラスタリングや異常検知などさまざまなものがあるが，中でもクラス分類によく使用されており，CDM の提唱者らも CDM の一つの有効性を示すために，クラス分類の結果を示している [Keogh 04b]．CDM はインスタンスとクラスの類似度ではなく，インスタンスとインスタンスの類似度を求めるものである．そのため，CDM を用いてクラス分類を行う際は，最近傍法などを用いて判定対象のインスタンスと，判定対象のクラスに属する全インスタンスとの類似度を計算する．クラス分類に

おいても，圧縮プログラムが頻出する部分文字列を特定し，その結果をもとにクラス分類を行っていることが，CDM によるクラス分類の特徴となっている．

CDM を計算する際に使用する圧縮プログラムには bzip2 を用いる．bzip2 には Burrows-Wheeler Transform [Burrows 94] が使用されており，情報量計算と同様，長い文字列の場合でも繰り返しを検知して圧縮できる．また，ZIP や GZIP などの辞書を用いる別の圧縮手法と比較して，正解率が高いという特徴がある．

CDM は教師なしで動作する類似尺度であるが，CDM と最近傍法を用いてクラスを推定する場合は，属するクラスが既知のデータを教師データとして使用し，属するクラスが未知のデータをテストデータとして使用する．

第 3 章

CDM の改良

3.1 圧縮プログラムで圧縮したあとのファイルサイズについて

圧縮プログラムを用いた類似尺度は，ファイルを圧縮した際のファイルサイズを利用している．しかし，圧縮後のファイルサイズにはファイル内部の情報を圧縮したデータだけではなく，そのデータを復元するための情報も含まれていると考えられる．実際に，何も情報が記述されていない 0 バイトのファイルを圧縮した時のファイルサイズは 0 ではない．

そこで，圧縮後のファイルサイズに含まれると考えられる情報が，類似度の推定に対してどのような影響を与えるかについて調査を行った．0 バイトから 100 バイトのランダムに生成された文字列に対し，BZIP2 を用いて圧縮した際のファイルサイズをプロットしたものを図 3.1 に示す．図において，横軸は圧縮前のファイルサイズであり，縦軸は圧縮後のファイルサイズである．この図に対して回帰直線を用いて圧縮後のファイルサイズに含まれる，元のデータを圧縮した以外の情報を推定すると，およそ 45 バイトであることがわかった．この値をオフセットとして CDM の計算時に差し引くことで，分類性能が向上することが明らかになっている [Takamoto 16]．このように，圧縮後のファイルサイズには元のデータには含まれていなかった情報が含まれており，その情報が分類に対して影響を与えていると考えられる．

3.2 CDM によるクラス分類の問題点とその対応

3.2.1 CDM の問題点

実際の活用においては，どの圧縮アルゴリズムを使用するかが，CDM を利用する際に重要な点となる．CDM は圧縮プログラムを利用し，その結果は圧縮プログラムに依存するためである．例えば，圧縮プログラムの実装において，分析する文字列の長さに上限がある場合，類似判定の対象となる文字列の長さが制限される．一方で，その長さの制限が

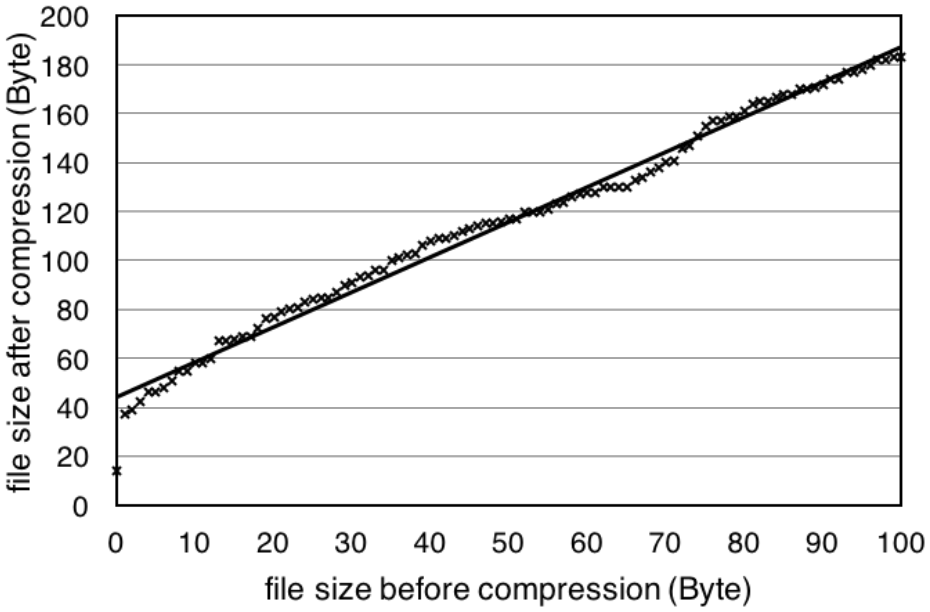


図 3.1: 圧縮前のファイルサイズに対する圧縮後のファイルサイズ

どのようなものであるかを，利用者が知ることは難しい．圧縮プログラムのアルゴリズムのなかには，実質的に長さの制限がない Block Sort 法を用いたものがあり [Burrows 94]，このアルゴリズムを用いた bzip2 は，長い文字列を判定の対象にする楽曲判定のタスクでの CDM の利用に適しているという報告がある [Takamoto 16]．

また，圧縮プログラムは類似度を判定するために作られたものではないことによる影響もある．圧縮プログラムで圧縮した際のファイルサイズは，入力文字列の情報量を反映したものであると述べたが，これは情報量そのものではない．例えば，“a” という文字列を圧縮したサイズは 39 Byte，“b” という文字列を圧縮したサイズは 39 Byte，“ab” を圧縮したサイズは 40 Byte であったとする．この時，CDM を用いて “a” と “b” の類似度を判定すると，その値はほぼ 1/2 となる．CDM において，同じ文字列同士の類似度を計算した場合，その値は 1/2 となる．しかし，上記の例で考えると，“a” と “b” は全く異なる文字列であるにもかかわらず，2 つの文字列は類似していると判定されることになる．実際に利用する場合では，文字列が十分に長ければこの問題は軽減されるが，対象の文字列が十分に長いかどうかは使用する圧縮アルゴリズムに依存する．

最後に，CDM をクラス分類に利用するために，CDM の提唱者はインスタンスが 1 つの最近傍法を用いたが，いくつのインスタンスを対象に判定するか，あるいは，そもそも最近傍法でよいのかを判定することも利用上の問題となる．

3.2.2　問題点への対応

CDM を用いたクラス分類の問題点を踏まえ，我々は以下のように CDM を改良する．まず，圧縮プログラムの出力が文字列の情報量を反映していることに着目し，特定の圧縮プログラムへの依存をなくすことを考える．我々は，圧縮後のファイルサイズの代わりとして，情報理論に基づいて定義した文字列の情報量を使用する．

T を圧縮対象の文字列とする．式 (2.1) の CDM の計算にでてくる $\mathrm{Comp}(T)$ は，圧縮プログラムを用いて T を圧縮した時のファイルサイズである．本研究では，これの情報理論に基づいた値を考える．まず，T をある方法で部分文字列の系列に分割する．これを $(\pi_1, \pi_2, \ldots, \pi_n)$ と書く．それぞれの π_k に対して，その出現確率 $P(\pi_k)$ に応じた長さの符号語を割り当てることを考える．この長さの合計は，圧縮結果のファイルサイズに相当する．情報理論では，π_k に割り当てる符号語の長さを π_k の自己エントロピー (選択情報量) と同じ値にすることで，最短の符号を得られることが知られている．そのため，T を $(\pi_1, \pi_2, \ldots, \pi_n)$ と分割して圧縮したときのファイルサイズの理論値は，π_k の自己エントロピーの総和となる．この理論値は分割方法に依存するが，我々は T をあらゆる方法で分割した値のうちの最小値を，分割方法に依存しない圧縮結果のファイルサイズの理論値と考えた．これを $I_s(T)$ と書くこととする．

圧縮プログラムは，圧縮対象の文字列自身の統計情報しか利用することができないが，$P(\pi_k)$ は，任意の文字列 T_s の統計情報を用いて計算することができる．これを $P(\pi_k; T_s)$ と書き，与えられた任意の文字列 T_s の中に文字列 π_k が出現する確率を表す．ここで T_s は確率を定めるパラメータとして作用するため，セミコロンを用いて表記する．$P(\pi_k; T_s)$ を用いて π_k の自己エントロピーの合計を考え，全ての分割での最小値を求めたものを $I_s(T; T_s)$ とする．相互情報量の一般的な記載方法である $I(X; Y)$ と類似しているが，これとは別物である．

最近傍法を用いたクラス分類にならうと， $\mathrm{Comp}(T)$ の理論値は $I_s(T; T)$ となるため，$\mathrm{Comp}(T)$ の代わりに $I_s(T; T)$ を用いると，クラス分類は次の式で表される．

$$\begin{aligned} &\mathrm{Class}(T) \\ &\quad = \arg\min_{C} \left\{ \min_{Y_i \in D(C)} \frac{I_s(TY_i; TY_i)}{I_s(T; T) + I_s(Y_i; Y_i)} \right\} \end{aligned} \tag{3.1}$$

ただし，C はクラスであり，$D(C)$ はクラス C に属していることがわかっている文字列の集合である．TY_i は文字列 T と Y_i を連結した文字列である．

Y_i がクラス C に属しているため，$D(C)$ に属する全ての文字列を結合した文字列 Y_C を考え，$I_s(Y_i; Y_i)$ の代わりに $I_s(Y_i; Y_C)$ を用いると推定がより正確になると考えられる．$I_s(TY_i; TY_i)$ は，T と Y_i を結合して圧縮したサイズに相当するが，Y_i は C に属しており，

式の計算時は T が C に属していると仮定していると考えられるため，$I_s(TY_i;TY_i)$ は $I_s(TY_i;Y_C)$ とするのが妥当である．よって，$I_s(TY_i;Y_C)$ と置き換えても良いと考えた．

$$\begin{aligned}&\mathrm{Class}(T)\\&\quad=\arg\min_C\left\{\min_{Y_i\in D(C)}\frac{I_s(TY_i;Y_C)}{I_s(T;T)+I_s(Y_i;Y_C)}\right\}\end{aligned}\tag{3.2}$$

さて，母空間に対応する文字列が共通のとき，文字列の情報量は以下のような分解を用いて近似できる．

$$I_s(T_1T_2;T_s)\sim I_s(T_1;T_s)+I_s(T_2;T_s)\tag{3.3}$$

CDM にならう方法として，情報量を Comp の代わりに使用する分類は最終的に以下のようになる．

$$\begin{aligned}&\mathrm{Class}(T)\\&\quad=\arg\min_C\left\{\min_{Y_i\in D(C)}\frac{I_s(T;Y_C)+I_s(Y_i;Y_C)}{I_s(T;T)+I_s(Y_i;Y_C)}\right\}\end{aligned}\tag{3.4}$$

ここで，$I_s(T;T)$ は判定するクラスに依存しないものであり，定数と考えるべきである．また，$I_s(Y_i;Y_C)$ は T に依存せず，その選び方に依存している．これは，判定に対するバイアスとなっており，無視すべきである．すると式 (3.4) は，$I_s(T;Y_C)$ が小さくなるクラスを選ぶことになる．言い換えれば，$I_s(T;Y_C)$ の大小による判定は，CDM によるクラス分類の改良となっている．

第 4 章

最大確率分割に基づいた情報量計算

4.1 最大確率分割情報量計算

一般に，文字列に含まれる情報量は文字列の生起確率から求められるため，文字列の情報量を推定するためには，その文字列が生成される確率を推定する必要がある．通常，文字列の情報量を求める場合，1 文字あたりの出現確率から文字の情報量を計算し合計することが考えられる．ある文字 c が持つ情報量 $I(c)$ は，c が出現する確率 p に依存する．したがって，文字列を 1 文字単位で構成される系列と考えると，長さ N の文字列 S に含まれる情報量 $I_c(S)$ は式のように計算できる．

$$I_c(S) = -\sum_{i=1}^{N} \log_2 P(c_i) \tag{4.1}$$

しかし，現実の文字列では，単語のように特定の部分文字列が 1 つのまとまりとして出現することが考えらえる．このことを考慮し，全ての部分文字列に対して生起確率を推定し，それをもとに文字列の情報量を推定する．

最大確率分割情報量では，対象の文字列をあらゆる方法で分割することを考え，分割によって得られる部分文字列の同時確率が最大になる，最も尤もらしいといえる分割方法を探す．情報量は，その最大化された同時確率から計算される．この際，文字列を分割して得られる部分文字列の生起確率を推定する必要がある．これについては，学習データのうちあるクラスに属するデータ群の中での，部分文字列の出現頻度から推定する．これにより，対象の文字列がそのクラスに属すると仮定したうえでの情報量を推定できる．以上を踏まえ，式 (4.2) の最大確率分割情報量 $I_s(T_1;T_2)$ を定義する．

$$I_s(T_1;T_2) = \min_{\pi_k \in \pi(T_1)} \left(-\sum_{t \in \pi_k} \log_2 P(t;T_2) \right) \tag{4.2}$$

ここで，T_1 は判定対象となる文字列 (テストデータ) を表している．また，$\pi(T_1)$ は T_1 に含まれる分割方法の集合であり，この集合の大きさは $2^{|T_1|-1}$ となる．ただし，$|T_1|$ は文

字列 T_1 の長さを表している．$\pi(T_1)$ に含まれるそれぞれの分割方法が π_k で表され，π_k に含まれる部分文字列が t で表される．T_2 は t が出現する母空間に相当する文字列，つまりクラスごとにまとめられた学習データを表している．ここで，$P(t;T_2)$ は，文字列 t の，T_2 中での出現確率を表しており，式 (4.3) のように推定する．

$$\hat{P}(t;T_2) = \frac{\mathrm{f}req(t;T_2)}{|T_2|} \tag{4.3}$$

式 (4.3) における $\mathrm{freq}(t;T_2)$ は, T_2 に t が出現した頻度であり, 分母の $|T_2|$ は T_2 の長さである．もし, T_1 の部分文字列 t が T_2 中に存在しない場合, その出現確率は $P(t;T_2)=0$ となり，T_2 における部分文字列 t 単体の情報量である $I_S(t;T_2)$ が計算できないが，その際は，$P(t;T_2)$ を，t が T_2 中に 1 度だけ出現した時の確率よりも小さな確率 (定数) として計算する．

4.2　分類における最大確率分割情報量計算

本節では，情報量計算を用いたクラス分類について説明する．式 (4.2) における T_1 をクラスが未知のデータ，T_2 を学習データをクラスごとに分けて結合したデータ群とする．この結合方法は同じクラスのデータを前のデータの後ろに連結する形で作成する．各クラスのデータ群は，同じクラスのデータをランダムに連結することでも作成できるが，頻度を計算する上では逐次連結した場合と大きな差はない．クラスごとに T_2 となるデータ群を作成し，T_1 の情報量を計算する．計算した結果，T_1 の情報量が最小 (最大の確率) となるものに用いた T_2 のクラスが推定結果となる．

判定に使用する情報量の計算方法を 4.1 に示す．未知の文字列 T_1 を “xyz” とすると，その分割方法は $2^{|T_1|-1}$ 通り存在するため，$\pi(T_1)$ =｛｛“xyz”｝, ｛“xy”, “z”｝, ｛“x”, “yz”｝, ｛“x”, “y”, “z”｝｝の 4 通りとなる．｛“x”，“yz”｝などの分割方法が π_k に相当する．T_1 に出現する部分文字列は “xyz”，“xy”，“yz”，“x”，“y”，“z” であり，これらが t に相当する．これらの部分文字列の出現頻度をクラスごとにまとめたデータ群から計算する．

4.1 の例であれば，クラス A のデータ群における部分文字列 “xy” の出現頻度は 0 である．一方，クラス B のデータ群における出現頻度は 3 であり，クラス C のデータ群における出現頻度は 2 である．これらを用いて，$\hat{P}(t;T_2)$ を計算する．各部分文字列の出現頻度を計算し，全ての組み合わせから情報量が最小となるものを選出する．情報量が最小となるものは，出現確率が最大となるものである．また，情報量はそれぞれの学習データ群を用いて計算されるため，クラス A のデータ群を元に計算された情報量とクラス B のデータ群を用いて計算された情報量，クラス C のデータ群を用いて計算された情報量の 3 種類が算出される．計算された 3 つの情報量のなかで最も小さいデータ群のラベルが未知

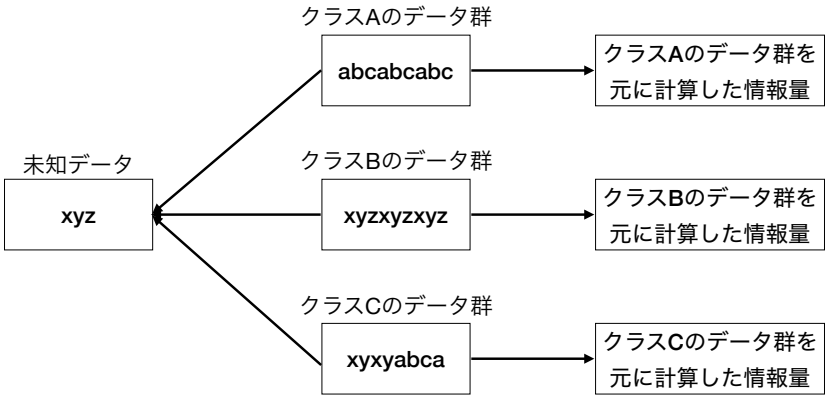


図 4.1: クラス分類における情報量の計算方法

データのラベルとなる.

第 5 章

最大確率分割情報量計算を用いた作曲者分類

5.1　本章の背景

音楽情報検索の分野において，楽曲の分類は基本的なタスクの一つであり，その中に楽曲の作曲者を分類するタスクがある．作曲者の分類タスクは，ある楽曲の作曲者が未知である場合に，その作曲者を推定し適切な作曲者クラスに分類するタスクである．1 人の作曲者によって作曲された楽曲の種類は多岐に渡る．クラシックの分野のみに着目しても，交響曲やオペラなど様々な種類があり，それぞれ異なる楽曲構造を持っている．これらの影響もあり，作曲者を分類するタスクは人間にもコンピュータにも難しいタスクである．コンピュータを用いた分類技術による知見は，作曲に関わる人々の音楽的な理解を促すだけでなく，音楽生成や推薦システムへの応用が期待できる．楽曲には様々な種類があるが，その中でも特にクラシック音楽の作曲者分類は，音楽の専門家たちによって幅広く議論されている．一方で，計算科学を応用した機械学習などの確率モデルによる分類は，人間の感覚では認識できないような音楽的パターンを捉える可能性があり，従来の分析を拡張したものと考えることもできる．したがって，確率モデルを用いて作曲者を分類しその特徴を分析することは，音楽の専門家に新たな視点を提供する可能性がある．

様々な作曲者分類手法が提案されているが，その多くは楽曲に含まれる様々な特徴量を抽出し，それを用いて分類器を学習する手法である．分類に特徴量を用いる手法では，特徴量の設計に高度な専門的知識が必要であり，設計に高いコストがかかる．一方で，専門的な知識が不要な手法を用いて作曲者分類に取り組んだ研究もある [Junior 12, Louboutin 16]．これらはテキスト分類によく使用される圧縮アルゴリズムを用いて分類を行っている．圧縮アルゴリズムを用いたテキスト分類は，深層学習などの手法と比較して，少ないリソースである程度の精度の分類ができるとして注目されている [Jiang 23]．しかし，圧縮アルゴリズムは 1 対 1 の比較しかできないため，分類に用いた場合はデータが増加するにつれ計算量が増加するという課題がある．そこで我々は，これらの手

法と同様に文字列の圧縮に基づくデータ分析手法で，学習データの増加にも対応した手法で，最大確率分割情報量の計算 [高本 23] を音楽分類タスクに適用する手法を提案する．

公平な手法の比較には，使用するデータセットの選定も重要である．例えば，異なる時代の作曲者を対象とした場合，適用した手法が作曲者の特徴を捉えているのか，時代の特徴を捉えているのかの判別が難しい．そこで，同じ時代に活躍した 2 人の作曲者である，ハイドンとモーツァルトのデータセットを用いて比較を行う．ハイドンとモーツァルトの類似性は長年議論されており，複数の先行研究で取り上げられている [Van Kranenburg 05, Hillewaere 10, Herlands 14, Kempfert 20]．同時代に活躍し，類似性がある作曲者の分類ができれば，各作曲者の特徴を分類器が捉えていると考えられる．

本章では，ハイドンとモーツァルトを分類対象として，提案手法の有効性を（1）圧縮原理に基づく分類手法との比較，（2）その他の機械学習手法との比較という 2 つの観点で評価する．

1 つ目の実験では，文字列化した楽曲に対し圧縮を利用して距離を計算する手法である，Normalized Compression Distance (NCD) [Li 04] をベースライン手法として比較を行った．NCD は音楽分類に広く使用されており [Cilibrasi 04, Cataltepe 05, Anan 12, Humphreys 21]，提案手法に用いている最大確率分割情報量と同様に圧縮の原理を応用していることからも比較対象として適している．実験の結果，ベースライン手法である NCD を用いた分類より，提案手法である最大確率分割情報量を用いた手法がより高い正解率を達成し，その差は統計的にも有意であることがわかった．

2 つ目の実験では，同じデータセットを用いて分類を行っている 4 つの先行研究 [Van Kranenburg 05, Velarde 16, Velarde 18, Kempfert 20] を対象に，正解率の観点から提案手法と比較した．比較に用いた 4 つの手法のうち，3 つの手法より提案手法が高い正解率を達成し，残りの 1 つともほぼ同等の性能が達成されたことがわかった．これらの結果を元に，提案手法が捉えている楽曲の特徴と，他の手法が捉えていると考えられる特徴との比較を行い，提案手法を用いる利点についても議論した．

なお，本章は Takamoto らの研究成果 [Takamoto 17] をもとに，複数の先行研究でも議論されている，2 人の作曲者の分類に焦点を当てたものである．[Takamoto 17] では，最大確率分割情報量を用いた分類を行っているものの，分割されたパターンについては注目していないが，本章では検出されたパターンの長さについても分析している．

5.2　関連研究

5.2.1　ハイドンとモーツァルト分類の難しさ

ハイドンとモーツァルトは 18 世紀（古典派時代）のほぼ同じ時期にオーストリアで活動しており，共にウィーン楽派の代表的な作曲家である．この 2 人の楽曲の分類は，専門知識を持った人間にも，機械学習でも困難なタスクである．人間による分類の難しさは，スタンフォード大学の Sapp らが作成している，非公式なオンラインクイズ[*1]の結果にも示されている．上記のサイトでは，自身の音楽的な経験レベルを自己申告し，実際にハイドンとモーツァルトの弦楽四重奏を聞いて，どちらの楽曲であるかを答えるものとなっている．最も正解率が高いのは，音楽についての経験レベルが 10 段階中 8 と申告したユーザで，その正解率は 67% 程度である．また，機械学習を用いて分類を行っているものとして，Van らの研究 [Van Kranenburg 05] がある．この研究では，複数のデータセットを用いて作曲者の分類を行っており，一つ抜き交差検証を用いた評価実験において，ハイドンとモーツァルトの分類における正解率が，79.44% と他の作曲者同士の分類と比較して低いことが明らかになっている．

上記の結果から，ハイドンとモーツァルトの楽曲分類は，人間にも機械にも難しいタスクであることがわかる．これは，2 人の作曲者が同じ時代に同じ地域で活躍しており，互いを尊敬しあう密接な関係にあったことなどの歴史的な背景が影響していると考えられる．よく似た作曲者である 2 人を高精度に見分けることができるのならば，時代や作風が異なる作曲者も見分けることができる可能性が高いと考えられる．したがって，本章ではハイドンとモーツァルトの分類タスクで手法を比較する．

5.2.2　作曲者分類

作曲者や楽曲の分類方法として，多くの手法が提案されている．これらの手法は主に，圧縮原理に基づく手法と，楽曲の特徴量を設計して用いる手法，ニューラルネットワークを用いた手法に分けられる．

圧縮原理に基づく手法として代表的なものは，Normalized Compression Distance (NCD) [Li 04] を用いる手法である．NCD は，圧縮プログラムを Kolmogorov 複雑性の近似として用いており，複数の先行研究で使用されている [Cilibrasi 04, Cataltepe 05]．Anan ら [Anan 12] は，様々な類似度と組み合わせて使用できる学習フレームワークを元に楽曲分類に取り組み，編集距離，NCD，Longest Common Subsequence，n-gram

[*1] http://qq.themefinder.org/ （参照：2023-09-15）

kernel，mismatch kernel の複数の類似度の性能を比較しており，NCD がもっとも高い分類性能をもたらしたことを報告している．Louboutin ら [Louboutin 16] は，NCD を元にした新しい尺度を提案し，LZ77，LZ78，Burrows-Wheeler，COSIATEC の 4 つの圧縮アルゴリズムを用いて分類の性能を比較しており，NCD に使用する圧縮プログラムによって性能が異なることを示している．Humphreys ら [Humphreys 21] は，文法ベースの圧縮プログラムを用いて NCD を計算することで，分類やエラー検出が可能であることを示している．これらの NCD を用いた分類では，その分類性能が使用する圧縮プログラムに依存することが示されており，圧縮プログラムの選択がハイパーパラメータとして作用している．

楽曲に含まれる特徴量を定義して分類を行う手法には，以下のようなものがある．Herlands ら [Herlands 14] は，global first-order, global higher-order, local first-order, local higher-order の 4 つのカテゴリ特徴量を定義し，ハイドンとモーツァルトの 2 人の楽曲を分類する手法を提案している．Kempfert ら [Kempfert 20] は，1182 個の特徴量から Bayesian information criterion によって有効な特徴量を選択し，ベイジアンロジスティック回帰を用いて，ハイドンとモーツァルトの分類タスクに取り組んでいる．上記の先行研究では，音楽的知識に基づいた特徴量を使用し，作曲者の分類を行っている．

また，近年の機械学習技術の発展に伴い，データの特徴を自動的に捉えるといえる，ニューラルネットワークが音楽情報処理の分野で広く活用されている．Velarde ら [Velarde 16] は，音楽中に出現する調の変化に対応するため，ピアノロール画像を用いた手法を提案している．Verma ら [Verma 19] は，楽譜画像の情報を元に特徴量の手動設計を排除したエンドツーエンドの分類器を作成し，2500 の楽譜データを用いて作曲者の分類を行っている．この方法は画像情報で訓練されたネットワークが必要である．Yang ら [Yang 21] は，GPT-2 を使用し，10 年間分のピアノコンクールの 200 時間を超えるデータから，楽譜と MIDI ファイルで学習した結果を用いて，One-shot による転移学習により，音声や MIDI データの分類を行っている．Walwadkar ら [Walwadkar 22] は，楽譜画像を用いた作曲家識別のためのエンドツーエンドディープニューラルネットワークモデルを提案した．32,000 の楽譜を使用して訓練し，9 人の作曲者の楽譜を用いて分類を行っている．Li ら [Li 23] は，音楽シーケンスの複雑さに対処するため，100 万曲を用いてトレーニングした改良型複合 Transformer モデルを提案し，音楽理解タスクで大きな性能向上を実現した．しかしながら，これらの手法は学習データが大量に必要で，作曲者の特徴が分析しにくいという問題がある．

我々は，特徴量設計が不要な作曲者分類手法として，圧縮原理に基づく手法に着目する．NCD は作曲者分類に広く用いられており，その信頼性が確立されている．最大確率分割情報量も圧縮原理に基づく手法であるが，広く調べられている作曲者分類における有効性はまだ明らかになっていない．本章では，公開されているデータセットに対して，最

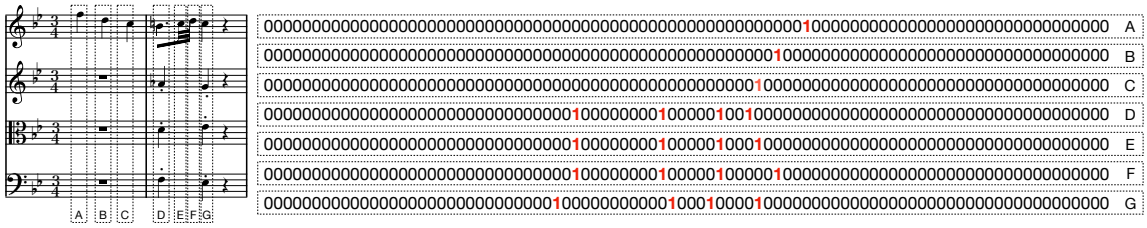


図 5.1: 文字列化前の楽譜と文字列化の例

大確率分割情報量を用いた分類を行う．

5.3　データセットと前処理

5.3.1　データセット

データセットは，複数の先行研究 [Van Kranenburg 05, Hillewaere 10, Taminau 10, Velarde 18] で使用されているデータセットと同様に，ハイドンとモーツァルトの弦楽四重奏を用いる．各弦楽四重奏は複数の楽章があり，それぞれは 2 つのバイオリンパートと，ヴィオラ，チェロのパートを合わせた，計 4 つのパートで構成されている．各楽曲は，音楽のデータ表現として一般的な MIDI データで表現されており，KernScores website[*2]から入手できる．録音などのオーディオデータには，演奏者の違いが反映される可能性があるため，MIDI データを用いる．このデータセットは，ハイドンの楽曲 54 曲とモーツァルトの楽曲 53 曲のほぼ同数の楽曲から構成された，計 107 曲のデータである．

5.3.2　楽曲の文字列化

提案手法や圧縮を用いたベースライン手法を楽曲の分類に適用するためには，楽曲を文字列化し計算可能な形にする必要がある．本実験では，楽曲の文字列化手法として，足達らの提案するクロマベクトルを 88 次元に拡張した文字列化方法を用いる [足達 13]．

まず，MIDI のノート情報を元に，音が切り替わったタイミングで再生されている音をサンプリングする．サンプリングしたタイミングごとに，音が再生されているときは 1，再生されていないときは 0 を各音へ割り当てることで，88 次元のベクトルを作成する．この手法はピアノ楽曲を前提としているため，ベクトルの次元数が一般的なピアノの鍵盤数である 88 次元となっているが，弦楽四重奏の音域も含まれているため，そのまま適用する．

以下に 5.1 の楽譜を文字列化する場合の例を示す．まず，各音符が出現している場所で

[*2] https://kern.humdrum.org/

サンプリングを行う．5.1 の例では，A から G の 7 点が音が切り替わるタイミングである．このとき，サンプリングされたタイミングで音が再生されている箇所を 1，それ以外の箇所を全て 0 とすることで 88 次元のベクトルを作成する．文字列化する楽曲に対し，サンプリングした各タイミングで文字列化を行い，前のデータの後ろに次のデータをそのまま連結することで，1 つの長い文字列を作成する（5.1 文字列部分）．これにより，楽曲内で基準となる音が変化する転調が発生しても，同じパターンとして認識できる．

楽曲の文字列化には，MIDI のノート情報から音の高さと長さを取り出して使用する．この文字列化では音の組み合わせと遷移が表現できる．音の長さの情報は音の遷移を取り出すために使う．長さの情報によって，音の変化のタイミングがわかる．音が変化したタイミングごとにサンプリングを行うため，リズムが異なるだけで同じ音の遷移からなる旋律は同じ文字列で表現される．また，音が変化したタイミングごとにサンプリングを行うため，スタッカートなどの表現は文字列へ反映されない．音の強さの情報も無視され，音が鳴っているかどうかだけ表現される．これは，同じ旋律で音の長さが異なるだけの場合を，同じ主題の変奏だと考えることとみなせる．

その他の楽曲の特徴として考えられるものに楽曲全体の構造があるが，我々は音の組み合わせとその遷移という，音楽的知識を必要としない分類に着目しているため，楽曲構造は利用しない．本研究で用いる文字列化では音の組み合わせがどのように遷移するかを表現するものとなっており，楽曲全体の構造は反映されない．

5.3.3　文字列化手法の利点と転調への対応

楽曲の中で基準となる調が変化する転調は，楽曲において重要な要素の 1 つである．ここでは，前処理として行っている文字列化が転調に対してどのように対応しているかを説明する．転調は，ある楽曲において使用されている和音の構成音が変動する形式である．この時，出現する旋律が同じものかの判定が必要になる．作曲者が好む旋律がある場合，調が異なるだけでそれらを違うものと判別するのは望ましくないためである．データとして MIDI を用いる場合，MIDI に含まれている音の高さを表すノート番号を使用する．このノート番号は，音の高さを表すために一番低い音から順に 0 から 127 までの番号を割り当てている．一般的な文字列化は，このノート番号を取得して連結するものが多い [Deepaisarn 23]．こういった手法では，楽曲中で調が変わった場合に異なるものと認識されるため，転調を抽出することができない．

こうした問題を解決するため，通常はどのような転調が起こる可能性があるかを踏まえ，転調を行った後の旋律との一致をはかる．別のアプローチとして，ピアノロール画像を用いた分析を行っているものがある [Velarde 16]．ピアノロール画像では，転調は画像の並行移動とみなすことができるため，転調しても一致する旋律を検出することができ

る．本研究で使用している文字列化により，楽曲は 1 次元の文字列として表現されている．また，各タイミングで文字列化した際に区切り文字などを利用していないため，相対的な音の組み合わせを取得できる．このため，音の組み合わせという特徴だけでなく，時間変化という特徴にも対応している．

5.3.4　音楽分類に対しての情報量計算

音楽分類へ最大確率分割情報量の計算を適用する方法について述べる．最大確率分割情報量は，文字列に含まれる情報量を部分文字列の出現確率に基づいて計算する手法であり，文字列 T_2 における文字列 T_1 の最大確率分割情報量 $I_s(T_1;T_2)$ は，以下の式で定義される．

$$I_s(T_1;T_2) = \min_{\pi_k \in \pi(T_1)} \left(-\sum_{t \in \pi_k} \log_2 \frac{\mathrm{freq}(t;T_2)}{|T_2|} \right) \tag{5.1}$$

ここで，T_1 は情報量を計算したい未知のデータを表し，T_2 は学習データである既知のデータを連結したものである．また，$\pi(T_1)$ は T_1 の文字列に含まれる分割方法の集合であり，この集合の大きさは $2^{|T_1|-1}$（ただし，$|T_1|$ は文字列 T_1 の長さ）となる．$\pi(T_1)$ に含まれる $2^{|T_1|-1}$ 個の各分割方法は π_k で表され，π_k に含まれる部分文字列が t で表される．部分文字列の出現確率は $\mathrm{freq}(t;T_2)$ から推定する．これは，判定対象となる文字列 T_1 に出現するあらゆる部分文字列 t の出現頻度を，母空間に相当する文字列である T_2 から計算することで，あるラベルの既知データの集合を元にして対象文字列の最大確率分割情報量を計算している．

以下に，最大確率分割情報量を楽曲に適用する例を示す．式 (5.1) における T_1 は，判定対象となる未知の楽曲である．T_2 は，既知データをクラスごとにまとめたもので，本実験の場合であればハイドンとモーツァルトの楽曲データをそれぞれ連結してまとめたものであり，T_2 は 2 個作成される．ハイドンの既知データをまとめた学習データセットを $T_2(\mathrm{Haydn})$，モーツァルトの既知データをまとめた学習データセットを $T_2(\mathrm{Mozart})$ とする．未知の楽曲 T_1 の例を 5.2 に示す．本来は 1 曲全てが T_1 となるが，簡単化のためにあるタイミングの和音のみで構成されるとする．この楽曲に含まれる分割の集合 $\pi(T_1)$ は 5.2 の (a) から (d) となる．また，(a)，(b)，(c)，(d) それぞれが π_k であり，分割された音符が t である．分割された音符 t を用いて T_1 の情報量を計算する．t の出現確率は，それぞれの学習データセットから計算される（5.3）．ハイドンの学習データセット $T_2(\mathrm{Haydn})$ に出現する全ての音符情報を用いて，全ての t の出現頻度から最大確率分割情報量を計算し，その値が最小になる理想的な分割方法を決定する（5.3 STEP 1）．同様の計算をモーツァルトの学習データセット $T_2(\mathrm{Mozart})$ でも行う．これにより，T_1 の最大確率分割情報量として，$T_2(\mathrm{Haydn})$ を元にしたものと $T_2(\mathrm{Mozart})$ を元にしたものの 2 つが得られ

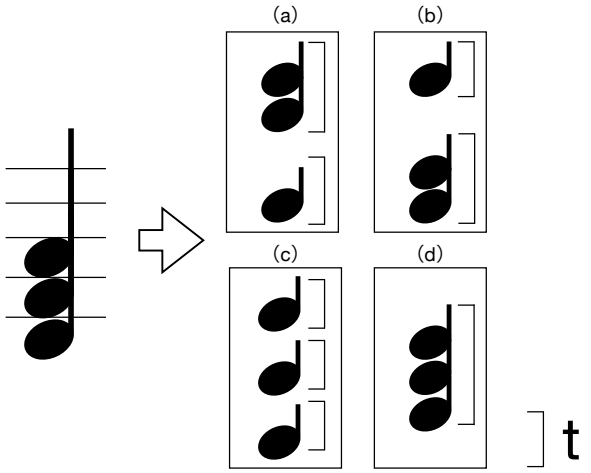


図 5.2: 楽譜の分割方法

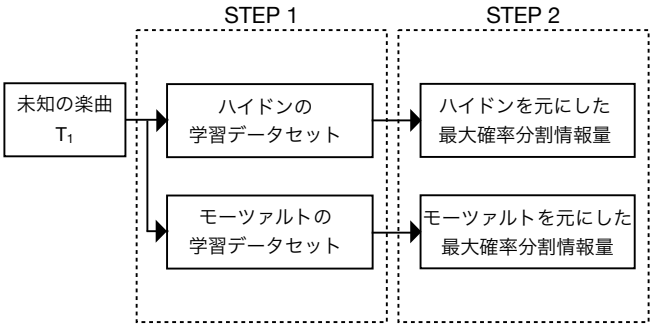


図 5.3: 最大確率分割情報量の計算方法

る（5.3 STEP 2）．最終的なテストデータ T_1 のラベルは，2 つの値でより小さい値を算出するのに使用した学習データセットのラベルとなる．

最大確率分割情報量は，T_2 で学習した時に T_1 が圧縮される上限と考えられる．全ての部分文字列の出現確率の計算は，計算量が実用的ではない定義式であるが，高本ら [高本 23] によってすでに十分に実用的な速度の高速化がされており，大規模なデータセットに対しても実際的な速度で使用できる．

5.3.5　最大確率分割情報量を用いた分類の生成モデルとしての解釈

x の情報量はあるクラスにおける未知データ x の出現確率を用いて計算する．本章における作曲者の分類は，ハイドンのクラス C_{Haydn} とモーツァルトのクラス C_{Mozart} のいずれかに未知データ x を分類するタスクである．通常の分類では，未知データ x がハイドンのクラス C_{Haydn} に属する確率 $P(C_{\mathrm{Haydn}} \mid x)$ と，モーツァルトのクラス C_{Mozart} に属する確率 $P(C_{\mathrm{Mozart}} \mid x)$ を比較する．一方で，学習に用いるデータセットでは $P(C_{\mathrm{Haydn}})$ と $P(C_{\mathrm{Mozart}})$ が同じ，すなわち均衡データであるという仮定をしている．したがって，$P(x \mid C_{\mathrm{Haydn}})$ と $P(x \mid C_{\mathrm{Mozart}})$ の 2 つを比較してクラスを決定する．これらの確率を求めることは，最大確率分割情報量を求めることと等価である．

5.4　NCD と最大確率分割情報量の比較

5.4.1　ベースライン手法

ベースライン手法として，提案手法と同様に圧縮の原理を利用した類似尺度である，Normalized Compression Distance (NCD) [Li 04] を実装する．NCD は，2 つの文字列間の距離を圧縮後のファイルサイズを用いて求めるものであり，以下の式で定義される．

$$\mathrm{NCD}(x, y) = \frac{C(xy) - \min(C(x), C(y))}{\max(C(x), C(y))} \tag{5.2}$$

ここで，x と y は文字列を表しており，xy は 2 つの文字列 x と y を連結したものである．また，$C(x)$，$C(y)$，$C(xy)$ はそれぞれの文字列を任意の圧縮プログラムで圧縮した後のファイルサイズを表している．もし，2 つの文字列が類似したパターンで構成されていれば，それぞれの文字列を個別に圧縮するよりも，連結して圧縮した方が圧縮率が高くなるという圧縮の原理が利用されている．この原理をもとに，2 つの文字列 x と y 間の距離（類似度）が計算される．NCD の値が小さいほど 2 つの文字列は類似しており，逆に 2 つの文字列が類似していなければ 1 に近づく．

楽曲の分類に NCD を用いる場合，式 (5.2) における x と y は，それぞれテストデータとなる楽曲 1 つと，学習データの楽曲の 1 つに相当する．ある楽曲のラベルを推定する場合，対象となる 1 曲と残り全ての学習データとの NCD をそれぞれ計算し，最も NCD の値が小さい学習データの作曲者を対象楽曲の作曲者と判定する（最近傍法）．また，NCD に用いる圧縮アルゴリズムは，ハイパーパラメータに相当するが，本実験では BZIP2 と gzip を用いる．

表 5.1: ベースライン手法と提案手法の正解数の比較

	正解数	正解率 [%]
NCD (BZIP2)	67/107	62.6%
NCD (gzip)	64/107	59.8%
提案手法	**89/107**	**83.1%**

5.4.2　評価方法

分類結果の評価には一つ抜き交差検証を使用する．一つ抜き交差検証は，ある 1 曲をテストデータとして選択し，それ以外の楽曲を学習データとする方法であり，データセット内の全ての楽曲（107 曲）に対して繰り返し行う．判定された作曲者のラベルがテストデータの作曲者と同じであれば正解とする．正解率は以下の式 (5.3) を用いて計算する．

$$\text{正解率} = \frac{\text{正解した楽曲の数}}{\text{データセットの総楽曲数（107 曲）}} \tag{5.3}$$

5.4.3　実験結果

本実験では，提案手法の有効性を検証するため，圧縮に基づく分類手法である NCD をベースライン手法として実装し，統計検定を用いて正解率を比較した．実験の結果を 5.1 に示す．2 つの圧縮アルゴリズムを用いたときの NCD の結果を比較すると，圧縮アルゴリズムによって正解数に差があり，BZIP2 の方が gzip と比べて性能が良いことがわかる．

一方，最大確率分割情報量を分類に適用した結果を，5.1 の提案手法の行に示す．2 つの圧縮アルゴリズムのうち，より性能の高かった BZIP2 を用いたときの結果と比較しても，提案手法の方が正解率が高いことがわかる．提案手法の結果と NCD (BZIP2) の結果について，統計検定を行った．検定には McNemer 検定 [岩崎 06] を用いる．McNemer 検定は，2 つの関連したカテゴリカル変数の比較に使用される統計的手法であり，ある変数が変化したときにもう一つの変数が変化するかを評価する．検定に必要な情報を 5.2 に示す．検定の結果，$p = 4.35 \times 10^{-6}$ となり，危険率 1% で統計的に有意な差があることがわかった．

表 5.2: 提案手法と比較手法における正解数

		提案手法		
		正解数	不正解数	合計
NCD	正解数	56	11	67
(BZIP2)	不正解数	33	7	40
	合計	89	18	107

5.4.4　考察

繰り返しになるが，圧縮の原理とは，類似したデータは個別に圧縮するよりも一緒に圧縮した方が圧縮率が高くなるというものである．NCD は，圧縮した後のファイルサイズから類似度を計算しており，圧縮の原理を用いている．提案手法も，[高本 23] で示されている通り，圧縮の原理を用いた最近傍推定と近似できるため，同様に圧縮の原理に基づいている．2 つの手法は同じ原理にしたがっているものの，その結果には差が現れている．

NCD の性能は使用する圧縮プログラムに依存するが，BZIP2 の正解数が gzip よりも高いことは圧縮アルゴリズムの観点から説明できる．gzip は，出現パターンをその出現順に検出して利用する方法であり，文字列を逆順に並べると圧縮後のサイズが変化するアルゴリズムである．一方，BZIP2 はブロックソートアルゴリズムを用いており，その圧縮サイズは文字列の出現順に依存しない．作曲者の分類タスクにおいては，パターンの出現順で類似度が変わる性質が悪影響を及ぼしている可能性がある．このため，gzip が BZIP2 と比較して性能が低くなっていると考えられる．

BZIP2 に使用されているブロックソートを使用した圧縮は，全部分文字列を分析することと等価な処理として知られており，提案手法の分割方法と類似している．BZIP2 を用いた NCD による作曲者分類手法と提案手法の違いは，NCD が曲と曲の 1 対 1 の類似度をもとに最近傍法を用いて作曲者のクラスを推定するのに対し，提案手法はクラスに属するデータ全体との類似度をもとに作曲者のクラスを推定するところである．サンプルが有限である場合，NCD のような 1 対 1 の比較をすると，偶然類似しているサンプルのクラスがどちらであったかという事実に影響を受ける．つまり，判定に用いる曲がどのように選ばれているかに依存する．一方，提案手法はクラス（ある作曲者）のサンプル全体に含まれるパターンから判定しているため，サンプルの選び方による影響が軽減されている．これらの理由から，最大確率分割情報量を用いた結果が，NCD よりも性能が良くなったと考えられる．

5.5　4 つの先行研究との比較

5.5.1　比較手法

楽曲の分類には様々な手法が提案されているが，手法の公平な比較は重要な課題である．ニューラルネットワークに代表される手法には様々なパラメータが存在するが，その決定には考慮しなければならないものが多い．そこで本節では，関連研究でも取り上げている手法のうち，同じデータセットを使用しており，現時点までに投稿されている 4 つの手法を用いる．比較に用いた手法は以下の 4 つである．

1. [Kempfert 20]：1182 個の音楽的な特徴量から効果的な特徴量セットを抽出し，ベイジアンロジスティック回帰を Iterative Conditional Minimization (ICM) で最適化して分類.
2. [Velarde 18]：スペクトログラムの画像解析を行い，k-nearest neighbour (kNN) を用いて分類.
3. [Velarde 16]：ピアノロール画像にガウシアンフィルタを適用し，得られたピクセルデータを線形判別分析を介して変換し SVM で分類.
4. [Van Kranenburg 05]：20 個の音楽的特徴をフィッシャーの判別分析を用いて変換し，kNN を用いて分類.

また，比較に用いる正解率は各論文に記載されている値を用いる．

5.5.2　比較結果と考察

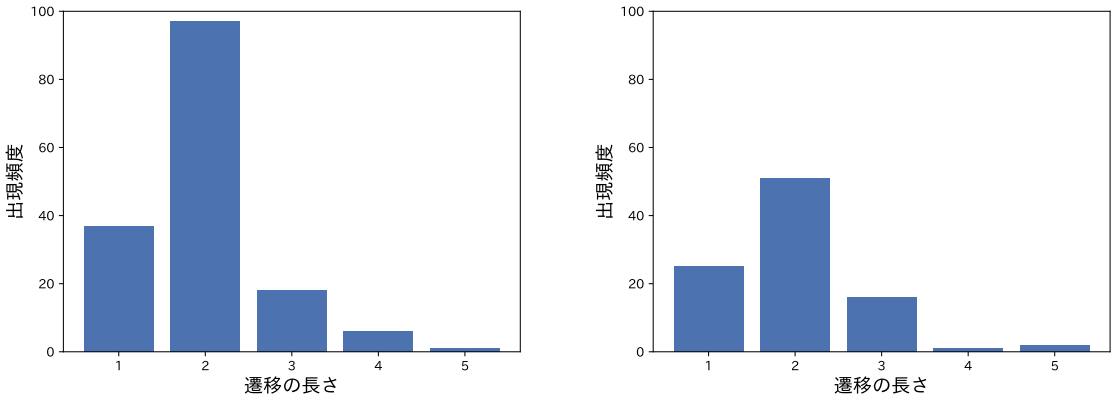
各手法の正解率と提案手法の正解率をまとめたものを 5.4 に示す．5.4 から，比較対象となる 4 つの手法のうち 3 つの手法で，提案手法がより高い正解率を示していることがわかる．特徴量抽出を必要としない提案手法が，その他の汎用的な機械学習手法と比較して正解率が上回ることは注目すべき結果である．

これに対して，提案手法より正解率が高い手法も存在するが，最大確率分割情報量を用いた手法には独自の利点が存在する（5.3）．Kermpfert らの手法 [Kempfert 20] は音楽的な知識に基づいて特徴量を設計した上で，古典的な手法を用いている．Kermpfert らが用いている特徴量の一つに，転調がある．実験に用いた弦楽四重奏はソナタ形式の楽曲であり，Kermpfert らの手法ではこのソナタ形式の楽曲における転調についての知識に基づいて特徴量を取得しているため，ソナタ形式に依存したものとなっている．一方提案手法では，前処理として行っている文字列化が特定の転調に依存しない頑健な設計となっており，最大確率分割情報量と組み合わせることでソナタ形式などの特定の形式に依存しない

表 5.3: Kempfert らの手法 [Kempfert 20] と提案手法の比較

	Kempfert らの手法	提案手法
転調への対応	ソナタ形式に依存	転調に頑健でありソナタ形式に依存しない
ベースの特徴	音楽知識を利用している	任意の長さの音パターンのみ
特徴の統合	ベイジアンロジスティック回帰	最大確率分割情報量

という利点がある．また，Kermpfert らの手法が転調以外の特徴量の抽出にも高度な音楽的知識を必要とするのに対し，提案手法は音楽的知識をほとんど必要とせず，任意の音のパターン情報のみを活用している．最後に，Kermpfert らはベイジアンロジスティック回帰という識別モデルを利用しているのに対し，我々の手法は最大確率分割によって生成確率を求めた上での比較であるため，生成モデルである．生成モデルは，他の作曲者を追加したときや，別の作曲者と分類を行うときであっても，同じモデルを利用することが可能であり一般性の高いアプローチである．



(a) モーツァルト　　(b) ハイドン

図 5.4: モーツァルトとハイドンの固有パターンの出現頻度

5.6　最大確率分割による遷移の長さの分析

本節では，最大確率分割情報量の計算を通じて求められた楽曲の分割パターンから，特徴として捉えられている音の組み合わせを分析する．特徴となるパターンは，以下のように抽出する．まず，テストデータ T_1 と同じラベルの学習データを使用して，最大確率分割情報量を計算し，理想的な分割を抽出する．テストデータ T_1 の理想的な分割は，式 (5.1) における $\pi(T_1)$ の中で，最も情報量が小さくなる（確率が大きくなる）ような分割

表 5.4: 提案手法と先行研究との正解率の比較

分類手法	正解率
（1）ロジスティック回帰（ICM）[Kempfert 20]	**84.1%**
（2）スペクトログラムの画像解析＋ kNN [Velarde 18]	74.8%
（3）ガウシアンフィルタ＋ SVM [Velarde 16]	80.4%
（4）フィッシャー判別分析＋ kNN [Van Kranenburg 05]	79.4%
提案手法	83.1%

表 5.5: ハイドンとモーツァルトの検出パターン数

	モーツァルト	ハイドン
総パターン数	28167	25316
重複なしのパターン数	21248	20150
固有パターン数	159	95

方法である．クラス分類では，情報量に着目していたが，分析ではその分割方法に着目する．ハイドンのある楽曲における理想的な分割とは，同じくハイドンの学習データを用いて計算されたものとする．これを理想的なパターンとして収集する．

各楽曲に対して分割を行った後，抽出したパターンの情報を 5.5 に示す．表において，パターン総数は，各作曲者の楽曲を分割した際に出現したパターンの総数を示す．そのうち重複を除いたものがパターン数（重複なし）の行に記載されている．さらに，重複のないパターンから片方の作曲者の楽曲のみに 5 回以上出現するものを選び出し，それを作曲者固有のパターンとする．この総数を固有パターン数の行に示す．

我々は，作曲者の特徴として，抽出された固有パターンの長さに着目した．文字列化によって，あるタイミングで再生されている音は 88 文字で表されている．そのため，固有パターンの長さを 88 で割ることで，特徴として捉えられた音の遷移の長さがわかる．出現する固有パターンの遷移の長さを作曲者ごとに計測した．その結果を 5.4 に示す．5.4 において（a）はモーツァルトの固有パターンの長さごとの頻度を，（b）はハイドンの固有パターンの長さごとの頻度を示している．遷移の長さが 1 となっているものは，パターンの長さが 1 文字以上 88 文字以下のものである．

5.4 より，固有パターンとして抽出された遷移の長さは長くても 4 程度であり，その大半は遷移の長さが 2 のものである．このことから，最大確率分割情報量はおよそ 1 から 4 の長さの遷移に相当する文字列を特徴として捉えている．特に遷移の長さが 2 のものが多いことから，ある音から次の音にどう遷移するかに着目していると考えられる．

固有パターンの総数を見ると，モーツァルトの固有パターンはハイドンの固有パターンの 2 倍近い数がある．すなわち，モーツァルトはハイドンより特徴的な遷移を多く使っていると考えられる．さらに，遷移の長さが 3 以上の固有パターンの割合がハイドンは 19.9% であるのに対して，モーツァルトは 15.7% である．このことから，ハイドンはモーツァルトに比べて長い固有パターンを持っていることが示唆される．

音の遷移を特徴として使用したいとき，その長さを適切に決定することは難しい．提案手法では，特徴となる遷移の長さは全部分文字列の出現確率によって自動的に決定される．圧縮に基づく手法は，任意の長さのパターンを考慮できるが，提案手法はその中からさらに特徴として捉えられているパターンを抽出できるという利点がある．

5.7　本章のまとめ

本章では，音楽分類のタスクにおいて，ハイドンとモーツァルトを対象とした作曲者分類を行った．我々は，文字列の最大確率分割情報量を音楽分類に適用する手法を提案し，同じく圧縮に基づく楽曲分類手法である，Normalized Compression Distance (NCD) と比較した．実験は，他の先行研究でも広く用いられているデータセットで，54 曲のハイドンの楽曲と 53 曲のモーツァルトの楽曲からなるデータセットを用いて行った．提案手法とベースライン手法である NCD の正解数を，一つ抜き交差検証で比較した．実験の結果，提案手法は NCD を用いた手法と比較して有意に優れた性能を示した．また，汎用的な機械学習手法を同一データセットに対して使用している 4 つの先行研究と比較を行い，音楽的知識を用いない提案手法が，音楽的な知識を用いて特徴量抽出を行う手法と同等またはそれ以上の正解数を達成することを示した．さらに，分類に利用された特徴的なパターンを用いて，その出現頻度と長さの観点から作曲者の特徴を分析した．

提案手法は，圧縮に基づいた分類手法が持つ利点を引き継ぎつつ，その欠点を解決している．提案手法や NCD のような圧縮に基づいた手法は，音楽的知識を用いた特徴量設計が不要であり，分類に使用されるパターンをその出現確率に基づいて自動的に決定するという利点がある．提案手法はその利点を引き継ぎつつ，ハイパーパラメータに相当する圧縮プログラムの選出が不要であり，NCD では困難であった 1 対多の比較や，分類に使用したパターンの抽出及び分析が可能である．

第 6 章

最大確率分割情報量計算を用いた時系列データ分類

6.1　本章の背景

ある未知のデータを適切なクラスに分類するデータ分類は，広く研究されているタスクである [Halvani 17, Micchi 18]．データを分類する際は，対象のデータと所属しているクラスが既知のデータとの距離を求め，その値をもとにクラスを分類する方法が一般的である．この距離を求める方法として，今までに多くの類似尺度が提案されている [Li 01, Li 04]．その中に，圧縮プログラムを利用した，Compression-based Dissimilarity Measure (CDM) という尺度がある [Keogh 04b]．CDM は，2 つの文字列が類似していれば，それぞれの文字列を個別に圧縮して合計したものよりも，連結して圧縮する方が圧縮後のファイルサイズが小さくなるという原理に基づいている．文字列を圧縮する処理は，頻出する文字列を特定し，頻度に応じてより短い文字列に置き換えることで，圧縮前の文字列の Byte 数に比べてより少ない Byte 数で同じ情報を表現できるものである．この CDM はさまざまな研究で利用されている [Cataltepe 07, Takamoto 16].

本章では，データのクラス分類を行うための類似尺度として CDM を使用するケースを想定し，その場合における CDM の改良方法を示す．改良方法は後の節で詳細を述べるが，CDM に使用されている圧縮プログラムの原理に着目し，文字列の情報量を頻度に基づいて求める手法を定式化したものを，高速化を含めて提案する．

さらに，CDM と提案手法によるデータ分類を行い，正解数と速度の観点から有効性を検証する．CDM の提唱者である Keogh らの論文 [Keogh 04b] では，CDM の分類性能の評価に時系列データを用いているため，同様に時系列データを用いることとする．本章では，まず [Keogh 04b] における分類タスクで使用されている 4 つのデータセットを用いて，CDM と提案手法を用いてクラス分類を行った．この実験から得られた正解数と，[Keogh 04b] に記載されているエラー率から正解数を算出した正解数を比較した．次に，同著者が公開している 128 個のデータセットに対して，同様に CDM と提案手法を

用いて判定を行い，その正解数を比較した．[Keogh 04b] との正解数の比較及び，多様なデータセットにおける正解数の比較の 2 つを用いて，提案手法の有効性を検証した．

同様に速度の面でも改良を行っている．文字列の出現頻度は，Suffix Array [Manber 93] によって実用的な速度で求められる．本章では Suffix Array を用いた頻度計数方法を工夫することで，情報量を計算する際に必要な，最大確率分割を求めるアルゴリズムの計算量を削減した．また，計算量を削減した情報量計算と CDM について性能の比較実験と同様に 128 個のデータセットで分類を行い，速度の観点から性能を比較した．上記の結果から，性能と速度両方の観点から CDM を改良できたことを示す．

6.2　関連研究

6.2.1　文字列間の類似尺度

SAX を用いてデータを文字列に変換することで，文字列間の類似度を図るさまざまな尺度を使用することができる．文字列間の類似尺度として代表的なものには，文字の置換や挿入，削除を行う操作の最小回数として定義される，レーベンシュタイン距離 [Levenshtein 66] やハミング距離 [Hamming 50]，n-gram の頻度に基づいた類似度などが存在する．これらの尺度は，CDM と同様に文字列間の類似度を計算するものであるが，その原理は CDM と根本的に異なっている．したがって，これらの距離と CDM を比較し，その差異について理論的な説明を行うことは難しい．一方で，情報量計算は，CDM に用いられている圧縮プログラムの原理に着目し導かれている．提案手法と CDM を比較しその理論的な背景について議論することは，性能の比較のみにとどまらず，文字列間の距離についてより本質的な考察が可能になることから，CDM との比較のみに着目している．

6.2.2　分類タスクに圧縮を利用した研究

CDM やそれに類似した尺度は，音楽の分類 [Cataltepe 07, Takamoto 16] にも用いられている．

1 つ目の研究 [Cataltepe 07] は，音楽のジャンル分類を行っている研究である．[Cataltepe 07] らは CDM と同様に圧縮プログラムを利用した尺度である Normalized Compression Distance (NCD) [Li 04] をさまざまな分類器と組み合わせることで高い分類性能を実現している．NCD は本実験で比較対象として用いる CDM と同様に圧縮後のファイルサイズを用いて類似度を計算する尺度であり，各データの圧縮後のサイズを考慮した類似度である．

2 つ目の研究 [Takamoto 16] では，音楽の分野に対して CDM を用いてピアノ楽曲の

作曲者の判定を行っている．[Takamoto 16] らは，圧縮プログラムに含まれる復元のために必要な情報がどの程度のサイズになるかを計算し，その値を圧縮後のファイルサイズから差し引くことで性能が向上することを示している．このことから，CDM の原理を忠実に使用することによって性能改善が見込めることが示唆されている．

6.3　時系列データの文字列化

情報量計算を用いたクラス分類も，CDM を用いたクラス分類も文字列に対して類似度を計算するものであるため，まず時系列データを文字列に変換する必要がある．本研究では，[Keogh 04b] にて報告されている方法に従い，Symbolic Aggregate Approximation (SAX) [Lin 03] を用いて時系列データを文字列に変換する．SAX は時系列データを構成する実数を適切な範囲に区切り，それぞれの区間を文字に対応させることで，サンプリング間隔ごとに 1 文字を生成し，実数の列を文字列に変換する方法である．

SAX を用いて時系列データを文字列に変換する場合，まず Piecewise Aggregate Approximation (PAA) と呼ばれるデータ圧縮作業を行う．長さ n の時系列データ C を w 次元の空間ベクトル $\bar{C} = (\bar{c_1}, \bar{c_2}, \ldots, \bar{c_w})$ に変換する場合，$\bar{C}$ の i 番目の要素は次の式を用いて変換される．

$$\bar{c_i} = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}\cdot i} c_j \tag{6.1}$$

これは，各データを w 個のフレームに分割し，フレームごとに平均値を取得することで，要素数を w 個に圧縮している．また c_j は時系列データ C の j 番目の要素を表している．正規化された時系列データが正規分布に従っていると仮定し，分布を各面積が等しくなるよう複数の領域に分けた後，各領域ごとにアルファベットを割り振る．データから計算された平均値がどの領域に属するか調べることで対応する文字が得られる．この処理を全てのデータ点に対して行うことで，時系列データを文字列に変換する．

6.4　CDM の論文と同一データセットによる比較

6.4.1　実験方法

この節では CDM の論文 [Keogh 04b] に掲載されているデータセットを用いて情報量計算の有効性を検証する．[Keogh 04b] と同様に Electrocardiogram (ECG) と Gun のデータセットをそれぞれ用意し，各データセットについて正解数を計算する．ここで正解数は Leave-one-out 交差検証を用いて算出する．データセットからある 1 つのデータをテストデータとして取り出し，そのデータを除く全てのデータを用いてテストデータのクラ

スを判定する．その結果，判定されたクラスがテストデータのクラスと一致していれば正解とし，それ以外を不正解とする．全てのデータに対して同様の処理を行い，最終的に正解と判定された個数を調べる．[Keogh 04b] では Euclidean Distance と Dynamic Time Warping (DTW) を CDM と比較しているが，[Keogh 04b] にて CDM と DTW の比較はすでに行われており，CDM の性能が最も高かったため，本実験では我々で再実装した CDM と情報量計算の正解数を比較する．

まず，ECG のデータセットについて説明する．ECG データセットは 4 人の患者 (chf01，chf05，chf10，chf15) の ECG データから構成され，BIDMC Congestive Heart Failure Database [Baim 86] から取得している．各患者のデータには 2 つの ECG 信号が含まれているため，それぞれの信号を Signal1 と Signal2 に分離し，1 次元の時系列からなる 2 つのデータセットを以下のように作成する．ECG データから連続した 3200 個のデータポイント (およそ 20 回の心拍) を取得する．それぞれの患者から 20 個ずつデータを抽出することで，80 個のデータを取得する．この処理を分離した 2 つの信号両方に行う．なお，SAX の変換に用いているパラメータは以下のとおりである．次元数 w はデータ数 n の半分である $n/2$ に設定し，変換に用いるアルファベット数は 10 文字とした．これらの各データに対して，あるデータをテストデータと仮定し，他の 79 個の学習データを用いてテストデータのクラスを判定する．

次に Gun のデータセットについて説明する．Gun のデータセット [Keogh 04a] は映像から抽出された時系列データである．2 人の俳優が銃を構えるか，単純に指をさす動作を行ったときの指の軌跡を抽出したものである．抽出したデータから連続した 1000 個のデータポイント（およそ 7 回の動作）を取得する．以下の 4 つのデータセットから 20 個ずつ，計 80 個のデータを取得する．

- A：俳優 1 が銃を構える
- B：俳優 1 が指をさす (銃を持たない)
- C：俳優 2 が銃を構える
- D：俳優 2 が指をさす (銃を持たない)

上記の 4 つのデータセットを以下のように分け，分類を行う．Gun 2 class では (A+B) と (C+D) の比較，つまり俳優 1 と俳優 2 の動作を分けるクラス分類を行う．Gun 4 class では，2 人の俳優と銃の有無を分ける 4 クラス分類を行っている．これらのデータセットについて，Leave-one-out 交差検証を行い，正解数を計算することで性能の比較を行う．

表 6.1: 各データセットにおける正解数

データセット名	CDM	CDM (実装)	情報量計算	データ数
ECG signal1	75	75	76	80
ECG signal2	74	77	77	80
Gun 2 class	80	77	78	80
Gun 4 class	76	75	76	80

6.4.2　実験結果

6.4.1 の実験について，CDM と情報量計算を用いてクラス分類を行った結果について，正解数を 6.1 に示す．6.1 について，“CDM” は [Keogh 04b] にて報告されたエラー率を正解数に換算したものを示しており，我々が CDM を再実装して求めたものが “CDM (実装)” である．6.1 を見ると，4 つのデータセットのうち 3 つで情報量計算の正解数が “CDM (実装)” より多くなっていることがわかる．また，残りのデータセットについても CDM と同等の性能を得ていることがわかる．

6.5　公開されているデータでの分類性能比較

6.5.1　使用したデータと実験方法

情報量計算が CDM よりも性能が良いことを示すために，[Keogh 04b] で用いられているデータセットよりも多くのデータセットに対しても比較を行う．Keogh らが同じく提供しているデータセットとして UCR Time Series [Dau 18] がある．このデータは必ずしも CDM で良い性能を示すものとは限らないが，多様な対象においての振る舞いを調査することは有用である．UCR Time Series に掲載されている 128 個のデータセットに対して，6.4.1 の実験と同様に CDM と情報量計算の比較を行う．これらのデータセットは，あらかじめ Dau らによってテストデータと学習データに分けられている．また，対象となるデータは数値で構成されているため，SAX [Lin 03] を用いて文字列に変換する．変換方法は 6.4.1 の実験と同様に，次元数 w を $n/2$ に設定し，変換に用いるアルファベット数は 10 文字とする．本来であればデータに合わせて SAX の次元数やアルファベット数を調整するべきであるが，本実験では CDM と情報量計算の差に対しての評価に着目しているため，一律で同じ次元数，アルファベット数を用いている．また，本実験における正解数は，判定結果として出力されたラベルとテストデータのラベルが一致したものを正

表 6.2: DTW より CDM のエラー率が低かったデータセットでの判定結果 (正解数)

データセット名	CDM	情報量計算
BeetleFly	15	14
BirdChicken	17	16
DistalPhalanxTW	83	81
Earthquakes	101	91
EthanolLevel	157	212
Ham	56	66
MiddlePhalanxOutlineAgeGroup	80	79
PigArtPressure	194	200
PigCVP	83	105
合計	786	864

解とし，その総数を用いる．

6.5.2　DTW より CDM の方が性能が良いデータセットでの実験結果

[Dau 18] に掲載されている 128 個のデータで CDM が効果的に機能しているデータを確認するために，時系列データの類似度を評価する方法として一般的な尺度である Dynamic Time Warping (DTW) [Müller 07] と比較を行う．なお，DTW でのエラー率については，Dau らが判定を行った結果が [Dau 18] に掲載されているため，本実験ではそのエラー率を基準として使用する．

CDM を用いて 128 個のデータを分類し，それぞれのエラー率を計算した結果，Keogh らが提示している DTW のエラー率を下回った（DTW より性能の良かった）データセットが 9 個あった．これらのデータに対して，情報量計算と比較した結果を 6.2 に示す．6.2 について，個別のデータセットで見ると，CDM より情報量計算の方が性能が良いデータセットは 4 個あることがわかる．一方で全体のデータで見ると情報量計算の方がより多く正解していることがわかる．この結果では CDM と情報量計算の判定性能については統計的に優位な差が得られなかったが，これは CDM と提案手法が類似の情報に着目して判定していることの帰結であると考えられる．

6.5.3　全てのデータセットでの実験結果

[Dau 18] に掲載されている全てのデータセットに対して比較を行った結果を 6.4 に示す．なお，6.4 において，最初の 5 列はそれぞれ、データセットの名前，テストデータのクラス数，テストデータの数，学習データの数，テストデータのデータ長 [byte] を示して

表 6.3: 各データの正解数

		提案手法		
		正解数	不正解数	合計
CDM	正解数	51 473	13 558	65 031
	不正解数	24 826	40 746	65 572
	合計	76 299	54 304	130 603

いる．なお，長さの値が Vary となっているものは，欠損などの理由でデータの長さが一定でなかったものである．これらのデータについては，Dau らが以下の方法で補完しており，本実験では補完されたデータを使用している．

長さが異なるデータの補完方法は以下のとおりである．データの欠測値は線形補間を用いて処理されており，各時系列の末尾に最も長い時系列の長さになるように低振幅のランダムノイズを付加することによって，長さのばらつきを調整している．

実験は 128 種類の時系列データに対し CDM と情報量計算を用いて分類した後，その結果の正解数を比較する．6.4 において “クラス数” の列は分類対象のクラス数を表しており，分類の難しさの推定に必要な情報である．また，“データ数” はテストに使用したデータの総数を表しており，正解率の推定に使用できる．“CDM(正解数)” の列はベースライン手法である CDM を用いて分類を行った場合の正解数であり，“提案 (正解数)” の列は情報量計算手法を用いて分類を行った場合の正解数を示している．

それぞれの手法を用いて判定を行った結果，ベースライン手法である CDM よりも正解数が向上したデータは，128 種類中 106 種類であった．また，この 128 個のデータセットで情報量計算が DTW より正解数が高かったデータは 21 個あった．

また，個別のデータで見た全てのデータにおける正解数を 6.3 に示す．6.3 において，CDM と情報量計算の両方で正解であったデータは 51473 個，CDM と提案手法の両方で不正解であったデータは 40 746 個，CDM のみ正解であったものが 13558 個，情報量計算でのみ正解であったものが 24826 個であった．改善したデータ数は改悪したデータ数の約 2 倍あり差は明らかであるが，McNemar 検定 [岩崎 06] を用いて性能の差を検定する．性能差がないことを帰無仮説とすると，危険率 9.91×10^{-732} で帰無仮説は棄却され，性能の向上は有意水準 1% で統計的に有意であることがわかる．この結果は CDM よりも広い対象で提案方法が優れていると解釈できる．

6.6　情報量計算手法における計算速度の改善

6.6.1　CDM と情報量計算の計算時間

先に述べたとおり，CDM を用いてクラス分類を行う場合，ある一つのテストデータと全ての学習データを比較する必要がある．一方で，情報量計算を用いてクラス分類を行う場合，前処理を行うことで学習データの大きさには依存しなくなるが，単純なアルゴリズムではテストデータの長さに対して複雑な計算が必要であり，かえって計算量が高くなる可能性がある．これは対象文字列の分割方法が，文字列長に対して指数関数的に増えるためである．一般的な圧縮プログラムは十分に高速であるため，たとえ全てのデータと比較を行わなければならないとしても，より複雑な計算をしている情報量計算に比べると高速である可能性がある．そこで，本節では情報量計算のアルゴリズムを高速化し，次章にて CDM と高速化した情報量計算の分類について，実行時間の観点から比較を行う．

6.6.2　SuffixArray を用いた出現頻度の計数

情報量の計算式を式 (4.2) で示される定義のまま使うことは現実的ではない．原理的に明快であり分類性能が良かったとしても，計算時間が桁違いにかかる場合は実用的な価値が小さいためである．本節では式 (4.2) を用いて計算される情報量と，同様の値を高速に計算することを検討する．

文字列の情報量計算には，Suffix Array [Manber 93] というデータ構造を用いるため，まず Suffix Array についての説明を行う．文字列 T_2 の Suffix Array は，$T_2\$$ の全ての接尾辞を辞書式順序でソートし，各接尾辞の位置を表す数値を格納した配列である．ここで \$ は終わりを示す文字である．また，\$ は T_2 中に出現しない文字であり，どの文字よりも辞書式順序で小さい文字であるとする．例として，$T_2 = \mathrm{abcabada}$ の Suffix Array を 6.1 に示す．Suffix Array 本体は図中央の数値配列（SA）であり，図右列（$T_2[\mathrm{SA}[i]\ \ldots]\$$）はその数値から参照される $T_2\$$ の接尾辞を表している．

T_2 の Suffix Array を事前に構築しておくことで，クエリとしてある文字列 t が与えられたとき，T_2 におけるクエリ t の出現頻度 $\mathrm{f}req(t;T_2)$ は，二分探索を用いることで $O(|t|\log_2|T_2|)$ で求められる．これを式 (4.3) に代入することで，クエリ t の出現頻度から出現確率を計算し，文字列の情報量を求める際に使用する．t の T_2 中での出現頻度は，「t を接頭辞とする T_2 の接尾辞の数」と言える．言い換えれば，T_2 の Suffix Array の各要素が示す文字列のうち、t から始まるものの数である．そのため，Suffix Array 上で t が接頭辞として出現する区間 $[sp, ep)$ が分かれば，出現頻度は $\mathrm{f}req(t;T_2) = ep - sp$ として求められる．ここで，sp は最初に Suffix Array 上で t が接頭辞として出現する場所を示

i	SA	T_2[SA[i] . . .] $
0	9	$
1	8	a$
2	4	abada$
3	1	abcabada$
4	6	ada$
5	5	bada$
6	2	bcabada$
7	3	cabada$
8	7	da$

図 6.1: $T_2 =$ abcabada の Suffix Array

	i	SA	T_2[SA[i] . . .] $
	0	9	$
	1	8	a$
sp →	2	4	abada$
	3	1	abcabada$
ep →	4	6	ada$
	5	5	bada$
	6	2	bcabada$
	7	3	cabada$
	8	7	da$

図 6.2: $T_2 =$ abcabada の Suffix Array で ab の出現頻度を求める様子

し，ep が接頭辞として出現しなくなる場所を示す．例として，$T_2 =$ abcabada の Suffix Array で ab の出現頻度を求める様子を 6.2 に示す．この場合，$[sp, ep) = [2, 4)$ であるため，ab の出現頻度は $4 - 2 = 2$ であることがわかる．

また，$[sp, ep)$ は，二分探索を用いることで $O(|t| \log_2 |T_2|)$ 時間で求められ，$freq(t; T_2)$ も $O(|t| \log_2 |T_2|)$ 時間で求められる．$O(\log_2 |T_2|)$ ではなく $O(|t| \log_2 |T_2|)$ となるのは，T_2 の接尾辞と t の大小判定に $O(|t|)$ 時間がかかるためである．

6.6.3　動的計画法を用いた情報量の計算手法

T_1 の T_2 をもとにした最大確率分割情報量 $I_s(T_1; T_2)$ は，T_1 のあらゆる分割方法を総当たりで計算すれば求めることができる．しかし，T_1 を分割する方法は $2^{|T_1|-1}$ 通り存在するため，$|T_1|$ の長さが大きくなると $I_s(T_1; T_2)$ を現実的な時間で求めることが困難になる．そのため，実際に $I_s(T_1; T_2)$ を求める際には，動的計画法を用いて計算量を削減する．

動的計画法は，対象の問題を小さな部分問題に分解し，各部分問題の計算結果を記録し

て，記録した結果を用いてより大きい問題を解いていくことを繰り返す手法である．この方法を情報量計算に応用すると，文字列の先頭から 1 文字ずつ計算する文字を増やしながら，そこまでの情報量を保存する DP 配列を作成する．文字を 1 つ増やすときには，DP 配列を利用して最小の情報量を計算する．分割の情報は，値を採用した配列の位置を記録することで求まる．この手法は，ビタビアルゴリズム [Viterbi 67] として知られているものである．詳細については，提案する計算手法の説明と合わせて述べる．

6.6.4　Suffix Array と動的計画法を組み合わせた高速な計算方法

高速化を行わない場合，$O(|T_1|^2)$ 回のループの中で $O(l \log_2 |T_2|)$ 時間 (l: クエリ長) の頻度計数処理を行う必要があり，全体で $O(|T_1|^3 \log_2 |T_2|)$ の時間がかかる．6.6.2 で述べたとおり，二分探索の際に文字列の比較を行うため，頻度計数処理にかかる時間がクエリ長 l に依存する．頻度計数処理を，文字列ではなく文字の比較のみで行えるのであれば，計数にかかる時間は $O(\log_2 |T_2|)$ となり，全体の計算量を $O(|T_1|^2 \log_2 |T_2|)$ に抑えられる．それを実現するため，頻度計数対象の文字列（T_1 の部分文字列）の規則性に着目し，計数の順番を工夫する．

情報量計算における動的計画法は以下のように実現する．計算結果を記録するための配列を $\mathrm{DP}[0 \ldots |T_1|]$ とする．高速化前の手法では $\mathrm{DP}[i]$ の定義を「T_1 の先頭から i 文字目までを切り出した文字列 $T_1[1 \ldots i]$ の情報量 $I_s(T_1[1 \ldots i]; T_2)$」としていたが，高速化後の手法では「$T_1$ の $i+1$ 文字目から末尾までを切り出した文字列 $T_1[i+1 \ldots |T_1|]$ の情報量 $I_s(T_1[i+1 \ldots |T_1|]; T_2)$」とし，高速化前と逆に，DP 配列の値を末尾から求めていく．こうすることで，先頭要素 $\mathrm{DP}[0]$ の値が $I_s(T_1; T_2)$ となる．例として，高速化前・高速化後それぞれでの漸化式を以下に示す．

高速化前の $\mathrm{DP}[i]$ は，$\mathrm{DP}[i] = I_s(T_1[1 \ldots i]; T_2)$ とし，$i = 0, 1, \ldots, |T_1|$ に対して以下のように計算する．

$$\begin{cases} \mathrm{DP}[0] = 0 \\ \mathrm{DP}[i+1] = \min\limits_{k=0 \ldots i} (\mathrm{DP}[k] \\ \qquad\qquad -\log_2 P(T_1[k+1 \ \ldots \ i+1]; T_2)) \end{cases} \tag{6.2}$$

高速化後の $\mathrm{DP}[i]$ は，$\mathrm{DP}[i] = I_s(T_1[i+1 \ \ldots \ |T_1|]; T_2)$ とする．そのため，$i = |T_1|, |T_1|-1, \ldots, 0$ に対して以下のように計算する．

$$\begin{cases} \mathrm{DP}[|T_1|] = 0 \\ \mathrm{DP}[i-1] \\ = \min\limits_{k=i \ldots |T_1|} (\mathrm{DP}[k] - \log_2 P(T_1[i \ \ldots \ k]; T_2)) \end{cases} \tag{6.3}$$

また，T_1 = aabd とした時の，DP 配列各要素の計算式を 6.3 に示す．図中では

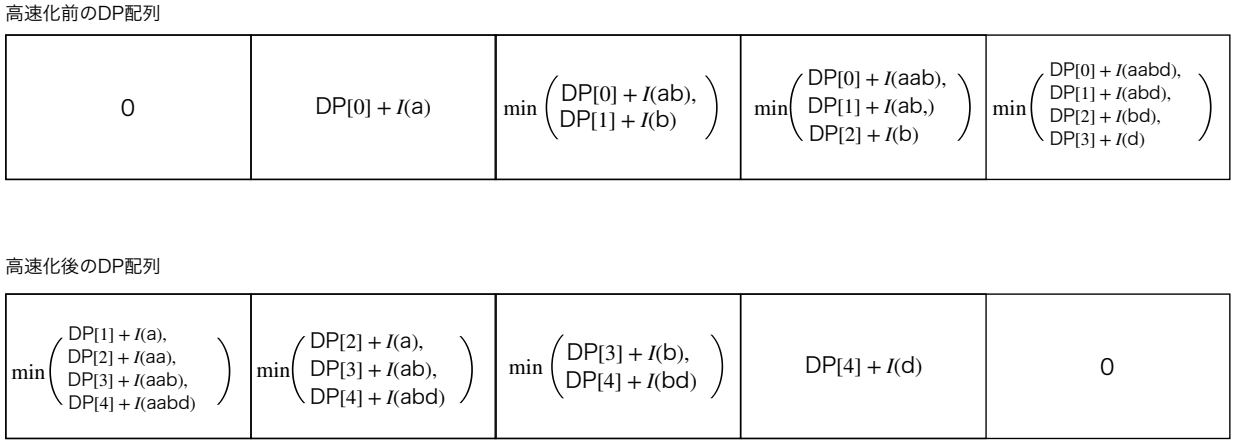


図 6.3: $T_1 = \mathrm{aabd}$ としたときの，高速化前・高速化後それぞれの DP 配列の比較．ただし $I(t) = -\log_2 P(t; T_2)$

$I(t) = -\log_2 P(t; T_2)$ と記載し，簡単化している．重要な点は，DP 配列の定義を変えたことで，各要素において計数対象となる文字列が高速化前のものと異なっていることである．例えば，高速化後の DP[1] を計算する際に計数対象となる文字列は {“a”, “ab”, “abd”} の 3 つである．これらは “a” という 1 文字の文字列から始まり，末尾に 1 文字ずつ文字を付け加えていった文字列の系列である．このような系列については，後述の手法を用いることで，各文字列の T_2 中での出現頻度を 1 つあたり $O(\log_2 |T_2|)$ 時間で求めることができる．これにより，$O(|T_1|^2)$ 回のループ中での頻度計数処理にかかる時間を $O(\log_2 |T_2|)$ に減らし，全体の時間計算量を $O(|T_1|^2 \log_2 |T_2|)$ に削減できる．

{“a”, “ab”,“abd”} などの，1 文字の文字列から始まり，末尾に 1 文字ずつ付け加えていった文字列の系列について，各文字列の T_2 中での出現頻度を 1 つあたり $O(\log_2 |T_2|)$ 時間で求める方法を説明する．ここでは例として，{“a”, “ab”,“abd”} の 3 つの文字列について，6.1 の Suffix Array を用いて頻度計数することを考える．

まず，“a” という文字列は長さが 1 であるため，二分探索によって $O(\log_2 |T_2|)$ 時間で計数できる．“a” は Suffix Array 上で $[1, 5)$ の範囲に出現することから，その出現頻度は $5 - 1 = 4$ であることがわかる（6.4）．次に，“ab” という文字列を計数することを考える．ここで注目すべきは，“ab” は “a” の末尾に 1 文字加えた文字列であるため，Suffix Array 上では必ず “a” の出現範囲の中にのみ出現する点である（6.5）．したがって，今回の場合では “a” の出現範囲 $[1, 5)$ のみを探索すればよい．さらに，“a” の出現範囲では 1 文字目が “a” であることが明らかであるため，文字列比較のために先頭 1 文字目を見る必要はなく，2 文字目に “b” が出現する範囲を求めれば，“ab” の出現範囲を求めたことになる．よって，二分探索時の文字列比較を探索する文字列の長さによらず一定で

i	SA	T_2[SA[i] . . .] $
0	9	$
1	8	a$
2	4	abada$
3	1	abcabada$
4	6	ada$
5	5	bada$
6	2	bcabada$
7	3	cabada$
8	7	da$

図 6.4: T_1 = abcabada の Suffix Array での a の出現

i	SA	T_2[SA[i] . . .] $
0	9	$
1	8	a$
2	4	abada$
3	1	abcabada$
4	6	ada$
5	5	bada$
6	2	bcabada$
7	3	cabada$
8	7	da$

図 6.5: T_1 = abcabada の Suffix Array での ab の出現

行えるため，“ab” の計数も $O(\log_2 |T_2|)$ 時間で行うことができる．“abd” についても同様に，“ab” の出現範囲で 3 文字目に “d” が出現する範囲を求めればよいため，これも $O(\log_2 |T_2|)$ 時間で計算できる．このように，一度計数した文字列の出現範囲を利用することで，系列中の全ての文字列頻度を $O(\log_2 |T_2|)$ 時間で計数できる．

6.7　計算速度の比較実験

6.7.1　使用するデータと実験方法

実験には 7 章で行った分類性能の比較実験と同様に，UCR Time Series [Dau 18] に掲載されている 128 個のデータセットを使用する．この 128 個のデータセットに対して，CDM と情報量計算を用いてクラス分類を行い，計算時間をそれぞれ計測し比較を行う．この時，情報量計算を使用した分類では，頻度の計算について 6.6.4 にて説明した手法を用いて計算量削減を行っている．計算時間は，time コマンドを用いて，プログラムの呼び出しから終了までにかかった実時間である “real” の値を計数する．この実時間には，Suffix Array の構築時間も含まれている．

6.7.2　実験結果

高速化後の情報量計算と CDM を用いた比較を 6.4 の “CDM（秒）” 列と “高速化（秒）” 列に示す．また，それぞれの列の値は CDM と高速化後の実行時間を表す．6.4 より，128 個のデータのうち 106 個のデータセットで情報量計算によるクラス分類が CDM よりも高速に動作した．CDM に用いている bzip2 の計算量を明言している文献を見つけられなかったが，おそらくデータ長に対してほぼ線形時間である．それに対し，情報量計算は時間計算量がテストデータ長の 2 乗に比例する．したがって，テストデータが長くなると情報量計算による分類にかかる時間は CDM より長くなる．しかし，テストデータの長さが固定であれば，学習データが多くなればなるほど，CDM より高速に計算できるようになる．

6.8　議論

6.8.1　なぜ性能が良くなったか

これを圧縮プログラムの動作に置き換えると，連続している文字列という制限はあるが，ある既知の情報をもとに対象のデータを最も効率よく圧縮した際のファイルサイズであると考えられる．CDM のように個々のデータから類似度を計算するのではなく，複数のデータから確率によって情報量を計算することで，より安定な値が得られると考えられる．

また，CDM の改良という点から考えると，4 章の式 (3) にて圧縮プログラムで計算される圧縮後のファイルサイズから情報量に変更したことで，ノイズとなりうる情報を削除できていると考えられる．そして式 (4) にて，個別のデータからではなく同じクラスに属するデータ全体から確率を計算するように変更している．その後式 (6) にて情報量を計算する際の依存先を特定し，余分な情報に依存しないよう変更している．また，学習データはそれが属するクラスから取り出されたものであり，あるクラスのサンプルと考えられる．最近傍法では 1 つのサンプルを選び，それによってクラスを推定するが，その選び方による分類結果への影響はノイズになると考えられる．この観点から，CDM のクラス分類では最近傍法を利用してきたが，最近傍だけではなく，複数の近傍を使用する k 近傍法を使うことも考えられる．k 近傍法を使用する際は，複数の近傍を判断のために選ぶことで，サンプルの選び方による影響を緩和できるが，そのための k の設定が恣意的であるとも考えられる．我々の提案手法では，サンプルの選び方によって推定結果に及ぼす影響を排除できており，かつ恣意的なパラメータを含むこともないと考えられる．これらの理由により，情報量計算を用いた分類が CDM を用いた分類よりも性能が良くなったと考えら

れる．

6.8.2　大規模なデータで計算時間はどのように変化するか

情報量計算は，可変長かつ任意長の文字列を対象に情報量をもとめるものであるが，バイグラムなどの固定長なモデル化に比べて，複雑で計算コストが高いと推測される．しかし，バイグラムに使用する 2 文字のパターンでは十分にパターンが取得できず，出現するパターン長の上限を予見することが困難である．このことから，文字列に出現する全ての分割を考慮することに意味がある．

本実験で用いたようなデータセットでは，学習データのサンプルは固定されているが，実際にクラス分類を行うシステムの実用的な場面を考えると，属するクラスが既知のデータが増えていくケースが多い．そのような場合は，増加したデータも学習データとして使用することが望ましい．このため，学習データの量とクラスの分類の速度との関係は重要である．

応用システムにおいては，時間をかけて学習データを前処理し，テストデータごとの判定時間を短くすることがしばしば行われる．提案手法では，学習データが増加した場合，それに比例して前処理の時間が増加する．一方で，テストデータごとの判定時間は，学習データの総サイズを N とすると，$O(\log_2 N)$ 時間となる．

6.9　本章のまとめ

本章ではデータのクラス分類タスクにおいて，圧縮を利用した代表的な尺度である CDM に着目した．しかし，CDM には以下の 2 つの問題点があった．1 つ目は，CDM に使用している圧縮プログラムが，復元のために不要な情報を保持している点である．2 つ目は，CDM が 2 つのデータ間の類似尺度であるため，全ての既知データに対して比較を行わなければならず，既知データが増加すると計算量も増加する点である．そこで本章では，これらの問題点に着目し，クラス分類タスクにおいて CDM を性能と速度の面で改良することを目的として，文字列の情報量を計算する方法を提案し，定式化した．

情報量計算を用いた分類では，前処理として既知データのクラスごとに，クラスに属するデータの全ての部分文字列の頻度を記録した表を用意しておく．判定対象データが与えられたら，そのありとあらゆる分割の中から，頻度表をもとに対象となる文字列が最も効率よく圧縮できる分割を選んだのち，それぞれのクラスごとに情報量を求める．そして，各クラスの情報をもとに計算した情報量の中から，値が最小になるクラスを選ぶ．これは，頻度表が与えられたときの情報圧縮の上限を求める方法であると解釈できる．また，CDM は利用する圧縮プログラムによって振る舞いが異なるのに対し，情報量計算は，数

式で量が定義されているため，その振る舞いについて再現性がある．

提案手法である情報量計算は，CDM に比べて速度面で劣る可能性があった．そこで，Suffix Array を用いた頻度計数方法と動的計画法を工夫して組み合わせることで計算量を削減した．その結果，高速化前は $O(|T_1|^3 \log_2 |T_2|)$ であった計算量を，$O(|T_1|^2 \log_2 |T_2|)$ に削減することができた．

また，実際的なデータにおける，提案手法の性能・速度面での有効性を検証するため，以下の実験を行った．性能の観点からは以下の 3 つの実験を行い，正解数を用いて性能を比較した．

CDM の提唱者である Keogh らの論文 [Keogh 04b] では，4 つのデータセットを用いて分類タスクでの CDM の有効性を検証している．1 つ目の実験では，Keogh らと同様のデータセットを取得し，CDM と情報量計算を用いて分類を行った．各データセットにおける正解数を比較したところ，4 つのデータセットのうち 3 つで正解数が向上し，残りの 1 つについても同等の正解数が得られた．

2 つ目の実験は，同じ Keogh らによって公開されている UCR Time Series [Dau 18] に掲載されている 128 個のデータセットを用いて行った．128 個のデータセットのうち，時系列データの類似尺度として有名な DTW よりも CDM を用いた分類の方が性能が優れていたデータセットが 9 種類あった．これらのデータセットに限定して CDM と情報量計算を比較したところ，4 つのデータセットで性能が向上した．また，データ全体で見ると，1650 個のデータのうち，CDM の正解数が 786 個であるのに対し，情報量計算では 864 個であり，情報量計算の方が正解数が多いことがわかった．

3 つ目の実験は，比較範囲を 128 個全てのデータに広げて行った．その結果，128 個のデータセットのうち，120 個のデータセットについて CDM より正解数が向上していることがわかった．データ全体で見ると，情報量計算を使うことで性能が低下するデータの個数よりも，性能が向上するデータの個数の方が 2 倍以上多かった．この結果に対して McNemar 検定を用いて統計検定を行ったところ，有意水準 1% で統計的に有意であることがわかった．上記の結果から，CDM が有効な対象については同様に有効でありながら，より多様なデータで効果があることがわかった．

速度の観点からは，性能の検証と同様に UCR Time Seires に計算されたデータセットに対して，各手法で分類を行うために必要な時間を計数し，比較を行った．実験の結果，128 個のうち 106 個のデータセットにおいて，CDM よりも提案手法の計算時間が短くなっていることがわかった．これらのことから，本章にて提案した文字列の情報量を計算する手法は，CDM を性能と計算量の両方で改良ができていることがわかる．

表 6.4: 提案手法と CDM を用いて分類を行った際の正解数と実行時間

データセット名	クラス数	テストデータ数	学習データ数	データ長 [byte]	CDM (正解数)	提案 (正解数)	CDM (s)	高速化 (s)
ACSF1	10	100	100	1460	33	54	2.18	20.96
Adiac	37	391	390	176	100	153	16.78	1.9
AllGestureWiimoteX	10	700	300	Vary	160	195	20.65	22.47
AllGestureWiimoteY	10	700	300	Vary	184	212	20.19	22.63
AllGestureWiimoteZ	10	700	300	Vary	170	228	21.54	20.3
ArrowHead	3	175	36	251	95	96	0.73	0.13
Beef	5	30	30	470	12	15	0.18	0.11
BeetleFly	2	20	20	512	15	14	0.72	0.46
BirdChicken	2	20	20	512	17	16	0.64	0.35
BME	3	150	30	128	93	96	0.4	0.53
Car	4	60	60	577	38	42	0.5	0.24
CBF	3	900	30	128	574	577	2.45	0.14
Chinatown	2	343	20	24	280	301	0.53	0.9
ChlorineConcentration	3	3840	467	166	2156	2256	182.4	1.51
CinCECGTorso	4	1380	40	1639	651	521	10.85	22.2
Coffee	2	28	28	286	24	26	0.13	0.25
Computers	2	250	250	720	145	158	7.54	3.8
CricketX	12	390	390	300	90	118	18.21	1.24
CricketY	12	390	390	300	90	108	21.36	0.98
CricketZ	12	390	390	300	83	111	17.89	1.14
Crop	24	16 800	7200	46	6009	8497	10476.59	7.76
DiatosizeReduction	4	306	16	345	233	239	0.54	0.39
DistalPhalanxOutlineAgeGroup	3	139	400	80	81	94	4.81	0.7
DistalPhalanxOutlineCorrect	2	276	600	80	188	197	12.6	0.1
DistalPhalanxTW	6	139	400	80	83	81	4.55	0.9
DodgerLoopDay	7	80	78	288	22	27	0.79	0.96
DodgerLoopGame	2	138	20	288	80	83	0.31	0.57
DodgerLoopWeekend	2	138	20	288	91	88	0.3	0.48
Earthquakes	2	139	322	512	101	91	6.41	0.37
ECG200	2	100	100	96	73	74	0.82	0.39
ECG5000	5	4500	500	140	3723	3768	211.52	1.98
ECGFiveDays	2	861	23	136	556	594	1.78	0.15
ElectricDevices	7	7711	8926	96	3228	3789	6309.64	7.52
EOGHorizontalSignal	12	362	362	1250	92	91	16.69	28.67
EOGVerticalSignal	12	362	362	1250	69	91	17.69	25.33
EthanolLevel	4	500	504	1751	157	212	32.88	44.51
FaceAll	14	1690	560	131	341	880	88.58	1.13
FaceFour	4	88	24	350	50	56	0.35	0.62
FacesUCR	14	2050	200	131	507	1078	36.58	1.1
FiftyWords	50	455	450	270	98	133	21.12	3.74
Fish	7	175	175	463	73	112	3.850	0.83
FordA	2	1320	3601	500	700	1013	824.14	3.93
FordB	2	810	3636	500	456	489	523.14	3.39
FreezerRegularTrain	2	2850	150	301	2170	2289	33.39	5.46
FreezerSmallTrain	2	2850	28	301	1902	2184	6.19	3.93
Fungi	18	186	18	201	75	124	0.22	0.45
GestureMidAirD1	26	130	208	Vary	50	52	2.37	2.78
GestureMidAirD2	26	130	208	Vary	37	48	2.32	2.38
GestureMidAirD3	26	130	208	Vary	25	22	2.28	2.3
GesturePebbleZ1	6	172	132	Vary	61	106	2.41	1.3
GesturePebbleZ2	6	158	146	Vary	50	92	2.48	1.28
GunPoint	2	150	50	150	119	129	0.54	0.64
GunPointAgeSpan	2	316	135	150	259	277	3.88	0.15
GunPointMaleVersusFemale	2	316	135	150	302	309	3.78	0.17
GunPointOldVersusYoung	2	315	136	150	242	262	3.34	0.18
Ham	2	105	109	431	56	66	1.43	0.15

表は次ページに続く

前ページからの続き

データセット名	クラス数	テストデータ数	学習データ数	データ長 [byte]	CDM (正解数)	提案 (正解数)	CDM (s)	高速化 (s)
HandOutlines	2	370	1000	2709	239	268	71.91	36.82
Haptics	5	308	155	1092	90	90	7.33	4.58
Herring	2	64	64	512	32	34	0.53	0.13
HouseTwenty	2	119	40	2000	107	97	0.96	2.91
InlineSkate	7	550	100	1882	116	142	8.16	34.51
InsectEPGRegularTrain	3	249	62	601	141	186	2.0	0.73
InsectEPGSmallTrain	3	249	17	601	134	147	0.53	0.53
InsectWingbeatSound	11	1980	220	256	367	442	45.11	5.11
ItalyPowerDemand	2	1029	67	24	742	870	4.85	0.39
LargeKitchenAppliances	3	375	375	720	204	284	11.720	14.58
Lightning2	2	61	60	637	41	40	0.47	0.26
Lightning7	7	73	70	319	21	31	0.49	0.19
Mallat	8	2345	55	1024	1532	2016	22.46	27.2
Meat	3	60	60	448	55	51	0.36	0.28
MedicalImages	10	760	381	99	352	370	23.97	0.49
MelbournePedestrian	10	2439	1194	24	979	1709	231.85	0.2
MiddlePhalanxOutlineAgeGroup	3	154	400	80	80	79	4.78	0.87
MiddlePhalanxOutlineCorrect	2	291	600	80	181	183	14.15	0.11
MiddlePhalanxTW	6	154	399	80	66	72	4.76	0.11
MixedShapesRegularTrain	5	2425	500	1024	1642	1665	645.0	27.14
MixedShapesSmallTrain	5	2425	100	1024	1525	1619	43.49	19.49
MoteStrain	2	1252	20	84	973	1081	2.13	0.79
NonInvasiveFetalECGThorax1	42	1965	1800	750	356	893	532.3	113.15
NonInvasiveFetalECGThorax2	42	1965	1800	750	492	1155	561.24	124.34
OliveOil	4	30	30	570	22	19	0.95	0.38
OSULeaf	6	242	200	427	103	150	6.67	0.6
PhalangesOutlinesCorrect	2	858	1800	80	539	564	138.75	0.4
Phoneme	39	1896	214	1024	289	260	106.4	30.93
PickupGestureWiimoteZ	10	50	50	Vary	14	18	0.217	0.32
PigAirwayPressure	52	208	104	2000	21	27	3.69	86.26
PigArtPressure	52	208	104	2000	194	200	7.6	17.52
PigCVP	52	208	104	2000	83	105	9.32	15.45
PLAID	11	537	537	Vary	87	37	29.91	249.42
Plane	7	105	105	144	81	105	0.97	0.79
PowerCons	2	180	180	144	150	164	2.88	0.73
ProximalPhalanxOutlineAgeGroup	3	205	400	80	156	157	7.36	0.88
ProximalPhalanxOutlineCorrect	2	291	600	80	199	216	15.18	0.12
ProximalPhalanxTW	6	205	400	80	144	149	9.49	0.1
RefrigerationDevices	3	375	375	720	151	185	21.56	1.32
Rock	4	50	20	2844	21	23	0.23	2.7
ScreenType	3	375	375	720	122	155	15.86	6.44
SemgHandGenderCh2	2	600	300	1500	394	335	64.77	1.86
SemgHandMovementCh2	6	450	450	1500	131	108	74.11	3.39
SemgHandSubjectCh2	5	450	450	1500	204	137	74.21	3.0
ShakeGestureWiimoteZ	10	50	50	Vary	17	26	0.26	0.29
ShapeletSim	2	180	20	500	86	124	0.54	0.98
ShapesAll	60	600	600	512	261	351	56.35	15.57
SmallKitchenAppliances	3	375	375	720	146	245	11.15	12.96
SmoothSubspace	3	150	150	15	57	77	1.44	0.6
SonyAIBORobotSurface1	2	601	20	70	358	419	0.84	0.26
SonyAIBORobotSurface2	2	953	27	65	543	706	1.92	0.38
StarLightCurves	3	8236	1000	1024	6506	6467	1175.62	116.68
Strawberry	2	370	613	235	329	335	26.29	0.6
SwedishLeaf	15	625	500	128	217	418	36.98	0.62
Symbols	6	995	25	398	707	570	2.64	1.69
SyntheticControl	6	300	300	60	92	172	7.52	0.45
ToeSegmentation1	2	228	40	277	139	171	1.31	0.12
ToeSegmentation2	2	130	36	343	64	86	0.56	0.98

表は次ページに続く

前ページからの続き

データセット名	クラス数	テストデータ数	学習データ数	データ長 [byte]	CDM (正解数)	提案 (正解数)	CDM (s)	高速化 (s)
Trace	4	100	100	275	87	84	0.86	0.27
TwoLeadECG	2	1139	23	82	801	904	2.26	0.93
TwoPatterns	4	4000	1000	128	1546	1991	394.77	1.52
UMD	3	144	36	150	119	123	0.46	0.56
UWaveGestureLibraryAll	8	3582	896	945	1057	1061	638.43	45.29
UWaveGestureLibraryX	8	3582	896	315	1012	1554	374.45	12.19
UWaveGestureLibraryY	8	3582	896	315	739	1120	363.73	13.37
UWaveGestureLibraryZ	8	3582	896	315	1041	136	371.6	12.28
Wafer	2	6164	1000	152	5909	5832	567.31	3.74
Wine	2	54	57	234	31	36	0.27	0.11
WordSynonys	25	638	267	270	150	181	22.24	2.86
Wors	5	77	181	900	39	44	2.59	0.73
WorsTwoClass	2	77	181	900	44	51	2.58	0.59
Yoga	2	3000	300	426	1993	2040	133.7	4.32

表 A.1 は以上

第 7 章

結論

本論文では，創造物に含まれる特徴を，データを生成するモデルの特徴と考え，これを計算機科学の観点から探究することを目的として，分類問題に取り組んだ．データ分類は，分類尺度がデータからなんらかの特徴を捉えて分類していると考えられるため，分類を通じて特徴が調査できると考えた．そこでデータの特徴を抽出する手法として，最大確率分割に基づく情報量計算を行う手法を提案した．提案手法は，圧縮を利用した類似尺度をもとにしている．圧縮を利用した尺度として代表的なものとして CDM や NCD などがあるが，これらの尺度はその特徴選択の詳細が不明瞭であった．提案手法である最大確率分割に基づく情報量計算手法は，文字列データ中に含まれる全部分文字列について，頻度情報をもとに情報量を計算することで，データの分類においてどの特徴が重視されているかが理論的に明らかにすることができる．

本論文では，音楽データにおける作曲者部類と時系列データ分類に取り組んだ．音楽データにおける作曲者分類は，音楽データに含まれる作曲者固有のパターンを特徴と考え，提案手法が分類及び分析に使用できるかを調査した．時系列データの分類については，様々な種類の時系列データを対象に，そのデータを生成したモデルの特徴を捉え，分類できるかを調査した．

音楽データにおける作曲者分類と時系列データ分類において，いずれも提案手法が圧縮を利用した尺度の性能を上回ることを示した．また，作曲者分類においては，作曲者の持つ特徴を頻度情報から実際に分析し，特徴となる長さについて示した．時系列データ分類においては，実際の応用を考慮し，分類性能だけでなく計算速度の観点から高速化を行い，オーダーの削減を行った．上記の結果から，全部分文字列の出現頻度をもとに文字列データの情報量を計算する提案手法は，データの分類および特徴分析が可能であることを明らかにした．

第 3 章では，最大確率分割情報量の計算手法が，圧縮を利用した類似尺度である CDM の改良になっている理論的な背景について述べた．第 4 章では，最大確率分割情報量の定義及び，それを用いたクラス分類ついて述べた．

第 5 章では，ハイドンとモーツァルトという 2 人の作曲者分類に対して，最大確率分割

情報量を適用した場合の結果について示した．ハイドンとモーツァルトは同時代，同地域において活躍した作曲者であり，その作風もよく似ていることから，専門家であっても分類が難しいとされている．加えて，楽曲に含まれる特徴は様々な種類があり，その中のどれが特徴として有効であるかは興味深い問題である．本研究で提案した最大確率分割情報量は，文字列化されたデータにおける全部分文字列の出現確率から計算されているため，純粋な音の組み合わせのみに着目したものである．最大確率分割情報量を用いて分類が可能であったという結果はすなわち，作曲者の特徴が瞬間的な音の組み合わせに強く現れる可能性を意味している．

また，同一データセットを用いて分類を行っている先行研究と比較しても同等かそれ以上の分類性能を発揮している．先行研究では多くの特徴量を設計する必要があるが，本手法を用いた分類ではそういった調整を行わずに分類が可能である．さらに，最大確率分割情報量は，その計算に使用した頻度の情報からデータの最も理想的な分割を求められることを明らかにした．

第 6 章では，最大確率分割情報量を時系列データに応用することを検討した．音楽データに出現するモチーフは，時系列データでも同様に出現すると考えられる．したがって本手法が有効に動作するのではないかと考えた．現実問題においても，時系列データは様々な種類があり，その分類は機械学習の技術を利用して積極的に取り組まれている重要なタスクである．本手法が時系列データ分類にも同様に機能すれば，最大確率分割を利用した情報量計算が，多くのデータに出現する特徴を捉えることができると考えられる．

この検証として，第 6 章では，公開されているデータセットを対象に先行研究と最大確率分割情報量計算を用いた分類の性能について比較を行った．実験の結果，多くのデータセットについて分類性能が比較手法よりも高いことが明らかとなった．また，情報量計算は全ての分割における部分文字列の出現頻度を計算する特性上，計算時間が課題であった．時系列データのように，データ自体の長さやその量が多いものについては，分類に係る時間を削減することも重要な課題である．第 6 章では，計算方法を工夫することで，実際的な時間で計算可能となる手法についても検討した．その結果，実際にオーダーを削減することに成功し，計算時間の削減も可能となったことを示した．

謝辞

本論文の執筆にあたり，指導教官である梅村恭司教授には終始温かいご指導を賜りました．また，北岡教英教授，渡辺一帆准教授には，本論文の審査を通じて適切なご助言を賜りました．深く感謝いたします．

神奈川大学工学部電気電子情報工学科の藤ノ木健介准教授にも，外部指導教員として様々なアドバイスをいただきました．心より感謝いたします．

梅村研究室の皆様にも多くのご支援・ご協力をいただきました．最後に，博士課程への進学とその生活を応援し支えてくださった，家族と友人に深く感謝いたします．

参考文献

[足達 13] 足達花絵, 岡部正幸, 梅村恭司 他：楽曲判定における和音の転回情報の効果, 研究報告音楽情報科学 (2013)

[Anan 12] Anan, Y., Hatano, K., Bannai, H., Takeda, M., and Sato, K.: Polypohic Music Classification on Symbol Data Using Dissimilarity Functions, in *International Society for Music Information Retrieval Conference* (2012)

[Baim 86] Baim, D. S., Colucci, W. S., Monrad, E. S., Smith, H. S., Wright, R. F., Lanoue, A., Gauthier, D. F., Ransil, B. J., Grossman, W., and Braunwald, E.: Survival of patients with severe congestive heart failure treated with oral milrinone, *Journal of the American College of Cardiology*, Vol. 7, No. 3, pp. 661–670 (1986)

[Burrows 94] Burrows, M. and Wheeler, D. J.: A block-sorting lossless data compression algorithm, Technical report, Digital Equipment Corporation (1994)

[Cataltepe 05] Cataltepe, Z., Sonmez, A., and Adali, E.: Music classification using Kolmogorov distance, in *Representation in Music/Musical Representation Congress* (2005)

[Cataltepe 07] Cataltepe, Z., Yaslan, Y., and Sonmez, A.: Music genre classification using MIDI and audio features, *EURASIP Journal on Advances in Signal Processing*, Vol. 2007, p. 036409 (2007)

[Cilibrasi 04] Cilibrasi, R., Vitányi, P., and De Wolf, R.: Algorithmic clustering of music based on string compression, *Computer Music Journal*, Vol. 28, No. 4, pp. 49–67 (2004)

[Dau 18] Dau, H. A., Keogh, E., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., Yanping, , Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G., and Hexagon-ML, : The UCR time series classification archive (2018), `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`

[Deepaisarn 23] Deepaisarn, S., Chokphantavee, S., Chokphantavee, S., Prathipasen, P., Buaruk, S., and Sornlertlamvanich, V.: NLP-based music processing for composer classification, *Scientific Reports*, Vol. 13, No. 1, p. 13228 (2023)

[Halvani 17] Halvani, O., Winter, C., and Graner, L.: Authorship verification based

on compression-models, *arXiv preprint arXiv:1706.00516* (2017)

[Hamming 50] Hamming, R. W.: Error detecting and error correcting codes, *The Bell system technical journal*, Vol. 29, No. 2, pp. 147–160 (1950)

[Herlands 14] Herlands, W., Der, R., Greenberg, Y., and Levin, S.: A machine learning approach to musically meaningful homogeneous style classification, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2014)

[Herremans 16] Herremans, D., Martens, D., and Sörensen, K.: Composer classification models for music-theory building, in *Computational Music Analysis* (2016)

[Hillewaere 10] Hillewaere, R., Manderick, B., and Conklin, D.: String Quartet Classification with Monophonic Models., in *ISMIR*, pp. 537–542 (2010)

[Humphreys 21] Humphreys, D., Sidorov, K., Jones, A., and Marshall, D.: An investigation of music analysis by the application of grammar-based compressors, *Journal of New Music Research*, Vol. 50, No. 4, pp. 312–341 (2021)

[岩崎 06] 岩崎 学：統計的データ解析入門　ノンパラメトリック法, 東京図書 (2006)

[Jiang 23] Jiang, Z., Yang, M., Tsirlin, M., Tang, R., Dai, Y., and Lin, J.: "Low-Resource" Text Classification: A Parameter-Free Classification Method with Compressors, in *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 6810–6828 (2023)

[Junior 12] Junior, A. D. D. C. and Batista, L. V.: Composer classification in symbolic data using PPM, in *2012 11th International Conference on Machine Learning and Applications*, pp. 345–350 (2012)

[Kempfert 20] Kempfert, K. C. and Wong, S. W.: Where does Haydn end and Mozart begin? Composer classification of string quartets, *Journal of New Music Research*, Vol. 49, No. 5, pp. 457–476 (2020)

[Keogh 04a] Keogh, E.: Index of / eamonn/SIGKDD2004 (2004), `http://www.cs.ucr.edu/~eamonn/SIGKDD2004`

[Keogh 04b] Keogh, E., Lonardi, S., and Ratanamahatana, C. A.: Towards parameter-free data mining, in *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 206–215 (2004)

[Levenshtein 66] Levenshtein, V. I., et al.: Binary codes capable of correcting deletions, insertions, and reversals, Vol. 10, No. 9, pp. 707–710 (1966)

[Li 01] Li, M., Badger, J. H., Chen, X., Kwong, S., Kearney, P., and Zhang, H.: An information-based sequence distance and its application to whole mitochondrial genome phylogeny, *Bioinformatics*, Vol. 17, No. 2, pp. 149–154 (2001)

[Li 04] Li, M., Chen, X., Li, X., Ma, B., and Vitányi P M, B.: The similarity metric,

IEEE transactions on Information Theory, Vol. 50, No. 12, pp. 3250–3264 (2004)

[Li 23] Li, Z., Gong, R., Chen, Y., and Su, K.: Fine-grained position helps memorizing more, a novel music compound transformer model with feature interaction fusion, in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 5203–5212 (2023)

[Lin 03] Lin, J., Keogh, E., Lonardi, S., and Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms, in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pp. 2–11 (2003)

[Louboutin 16] Louboutin, C. and Meredith, D.: Using general-purpose compression algorithms for music analysis, *Journal of New Music Research*, Vol. 45, No. 1, pp. 1–16 (2016)

[Manber 93] Manber, U. and Myers, G.: Suffix arrays: a new method for on-line string searches, *siam Journal on Computing*, Vol. 22, No. 5, pp. 935–948 (1993)

[Micchi 18] Micchi, G.: A neural network for composer classification, in *International Society for Music Information Retrieval Conference (ISMIR 2018)* (2018)

[Müller 07] Müller, M.: Dynamic time warping, *Information retrieval for music and motion*, pp. 69–84 (2007)

[Pollastri 01] Pollastri, E. and Simoncelli, G.: Classification of melodies by composer with hidden Markov models, in *Proceedings first international conference on WEB delivering of music. WEDELMUSIC 2001*, pp. 88–95IEEE (2001)

[Takamoto 16] Takamoto, A., Umemura, M., Yoshida, M., and Umemura, K.: Improving compression based dissimilarity measure for music score analysis, in *2016 International Conference On Advanced Informatics: Concepts, Theory And Application* (2016)

[Takamoto 17] Takamoto, A., Yoshida, M., Umemura, K., and Ichikawa, Y.: Computing information quantity as similarity measure for music classification task, in *2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications*, pp. 1–6 (2017)

[高本 23] 高本綺架, 小原佑斗, 吉田光男, 梅村恭司：データ分類タスクにおける Compression-based Dissimilarity Measure の精度と速度の改良, 人工知能学会論文誌, Vol. 38, No. 1, pp. A–M71_1 (2023)

[Taminau 10] Taminau, J., Hillewaere, R., Meganck, S., Conklin, D., Nowé, A., and Manderick, B.: Applying subgroup discovery for the analysis of string quartet movements, in *Proceedings of 3rd international workshop on machine learning and music*,

pp. 29–32 (2010)

[Van Kranenburg 05] Van Kranenburg, P. and Backer, E.: Musical style recognition —a quantitative approach, in *Handbook of pattern recognition and computer vision*, pp. 583–600, World Scientific (2005)

[Velarde 16] Velarde, G., Weyde, T., Chacón, C. C., Meredith, D., and Grachten, M.: Composer recognition based on 2d-filtered piano-rolls, in *International Society for Music Information Retrieval Conference*, pp. 115–121 (2016)

[Velarde 18] Velarde, G., Cancino Chacón, C., Meredith, D., Weyde, T., and Grachten, M.: Convolution-based classification of audio and symbolic representations of music, *Journal of New Music Research*, Vol. 47, No. 3, pp. 191–205 (2018)

[Verma 19] Verma, H. and Thickstun, J.: Convolutional composer classification, in *International Society for Music Information Retrieval Conference* (2019)

[Viterbi 67] Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, *IEEE Transactions on Information Theory*, Vol. 13, No. 2, pp. 260–269 (1967)

[Walwadkar 22] Walwadkar, D., Shatri, E., Timms, B., Fazekas, G., et al.: CompId-Net: Sheet Music Composer Identification using Deep Neural Network, in *4 th International Workshop on Reading Music Systems*, p. 9 (2022)

[Yang 21] Yang, D. and Tsai, T.: Composer Classification With Cross-Modal Transfer Learning and Musically-Informed Augmentation., in *ISMIR*, pp. 802–809 (2021)

博士論文に関する論文

本論文における第 3 章の内容は次の査読付き国際会議論文として公表済みである.

- Ayaka Takamoto, Mitsuo Yoshida, Kyoji Umemura.
  Analysis of Home Location Estimation with Iteration on Twitter Following Relationship.
  The 2016 International Conference on Advanced Informatics: Concepts, Theory and Application (ICAICTA 2016), Penang, Malaysia, August 2016.

本論文における第 6 章の内容は次の査読付き学術雑誌論文として公表済みである.

- 高本 綺架, 小原 佑斗, 吉田 光男, 梅村 恭司.
  データ分類タスクにおける Compression-based Dissimilarity Measure の精度と速度の改良.
  人工知能学会論文誌, Vol. 38, No. 1, pp. A-M71_1 (2023).

本論文における第 5 章の内容は次の査読付き学術雑誌論文として公表済みである.

- 高本 綺架, 廣中 詩織, 梅村 恭司.
  圧縮原理に基づく最大確率分割情報量を用いた作曲者分類.
  人工知能学会論文誌, Vol. 39, No. 2, pp. F-NA1_1 (2024).