令和四年度　特別研究報告書

環境音を用いた人の動作推定に関する検討

龍谷大学大学院　理工学研究科　情報メディア学専攻

T19M057　北迫祐樹

指導教員　三好力　教授

内容梗概

音声認識による操作には同じサービスを受けるために同じ音声を繰り返す必要があるという不便な点がある.

本研究では,マイクロフォンを内蔵し音声認識による操作を目的とした入力デバイスを利用し,環境音を用いて人の動作を推定するシステムの提案を行った.特徴的な環境音をシンボル音として分類し,ユーザーの家に設置された複数台の音声 AI が搭載された機器をつかい,命令やシンボル音の種類と発生した時間とを記録し,収集したデータを用いることで,特定のシンボル音の並びが類似した時間間隔で発生する度に特定の命令が行われているといったパターンを分類することが可能になる.頻度の高い特定の命令の前に発生したシンボル音のパターンとそれに類似したシンボル音の発生パターンを認識したとき,特定の命令を行うことをユーザーに提案する.

システムの実現に必要な検証として特定のシンボル音の識別が可能であるか,環境音をスペクトログラム画像に変換し畳み込みニューラルネットワーク(CNN)の学習と識別で検証した.次に識別により収集したデータから特定のシンボル音のパターンが分類可能であるか,録音データを基に作成した検証用のダミーデータ列を使い主成分分析による次元削減と k-means クラスタリングで検証した.

Abstract

The operation by voice recognition has an inconvenient point that it is necessary to repeat the same voice to receive the same service.

In this study, we proposed a system that uses environmental sounds to estimate human actions by using an input device with a built-in microphone for voice-recognition-based operations. By classifying characteristic environmental sounds as symbolic sounds, and using multiple voice AI-equipped devices installed in a user's home to record the types of commands and symbolic sounds and the time at which they occur, the collected data can be used to classify patterns such as "a specific command is given every time a sequence of specific symbolic sounds occur at similar time intervals. By using the collected data, it is possible to classify patterns such as "a specific command is given every time a specific sequence of symbol sounds occur at a similar time interval". When a user recognizes a pattern of symbol sounds occurring before a specific command with a high frequency and a pattern of symbol sound occurrences similar to it, the system proposes that the user perform a specific command.

To verify whether the system is capable of identifying specific symbol sounds, environmental sounds were converted into spectrogram images and verified through convolutional neural network (CNN) training and identification. Next, we verified whether it is possible to classify specific symbolic sound patterns from the data collected by identification, using a dummy data sequence created for verification based on recorded data, by dimensionality reduction using principal component analysis, and k-means clustering.

目次

第1章 はじめに

近年スマートフォンをはじめ,スマートスピーカー, IoT 機器といったマイクロフォンを内蔵し音声認識による操作を目的とした入力デバイスとして機能する機器が普及している[1][2].しかし,音声認識による操作には同じサービスを受けるために同じ音声を繰り返す必要があるという不便な点もある.音声操作によって照明をつける状況を想定したとき,毎日のルーティンの中で何度も照明をつけるために同じ命令を繰り返すのは不便である.現状ではユーザーからの命令を待ち続ける受動的な動作しか行わず,ユーザーの状況に合わせたサービスの提案や提供といった能動的な動作は行うことができない.

そこで,音声認識による操作を目的とした入力デバイスに搭載されているマイクロフォンを使い,同じサービスを受けるために同じ音声操作を繰り返す必要を段々と省略することが可能なシステムがあれば,既に普及したデバイスを普段通りに利用し続ける中で,マイクロフォンが収集する音を使いユーザーの状況を推測し共有することが可能になり,ユーザーの状況に合わせたサービスの提供だけではなく,室内見守りシステムの構築など幅広い分野への応用が期待される.

本研究では,マイクロフォンを内蔵し音声認識による操作を目的とした入力デバイスを利用し環境音を用いて人の動作を推定するシステムの提案, システムの実現に必要な特定の環境音の識別が可能であるかの検証,識別し記録された特定の環境音のパターンが分類可能であるかについて検証を行う.

本論文の構成は次の通りである. 本章では, 本研究が環境音を用いて人の動作を推定するシステムを提案することで, 同じサービスを受けるために同じ音声を繰り返す必要があるという音声操作の不便さの解消を目的とすることを述べた. 2 章では, 既存研究として, マイクロフォンを用いた音声以外の環境音に関する研究と既存技術の問題点, 環境音の識別に使用する畳み込みニューラルネットワークや環境音の発生パターンの分類に使用するクラスタリング手法についての説明を行う. 3 章では提案システムの概要と, 検証すべき課題として環境音同士の識別が可能か,

可能であれば目的とする環境音の発生パターンの分類が可能であるかについて述べる.4 章では二つの課題を検証すべくそれぞれに対して行った機械学習による識別とクラスタリングの実験および結果について述べる.5 章ではそれぞれ２つの実験結果に対する考察,そして全体の考察としてシステムの実現性が検証されたことについて述べる.6 章では本論文のまとめと今後の展望について述べる.

第2章 既存研究

2.1 スマートスピーカー

スマートスピーカー[1]とは，音楽再生の機能と対話型の音声操作に対応した AI サーバと通信を行う機能を持つスピーカーのことである.内蔵されているマイクロフォンで.スマートフォンや一部の IoT 機器[2]と同じく Wi-Fi，Bluetooth 等の無線通信を経由しメーカーが提供する AI アシスタントのサーバと Peer to Peer のネットワークを形成する.サーバ側で自然言語処理された音声認識結果をディープラーニングにより応答の最適解を常に機械学習し,最適なタスクの実行や発話で入力を行ったデバイスからの応答を可能とする.これにより,これまで PC 等の端末で行っていたキーやタッチによる入力操作を音声で補完することや,サーバにインターネットで接続しているさまざまなプラットフォーム上のサービスの利用,インターネットに接続された家電の操作,無線リモコンの代替操作を音声で行うホームオートメーション,自動車の車載機に搭載しての制御等の幅広い利用が可能になっている.ユーザーからの発話の全てに反応しないために,収集する音の中からウェイクワードと呼ばれる各メーカーが決めた特定の単語を認識した後,続く音声を命令として認識するようになっている.

2.1.1 Amazon Echo

Aamzon Echo は, Amazon が提供するスマートスピーカーのことである.実際の写真を 図 2.1.1 に示す.図からわかるとおりに側面がスピーカーとマイクで構成されており,ユーザーからの発話を音声アシスタントサーバーへ経由する端末である.Wi-Fi 接続 が可能な環境であれば専用のアプリをダウンロードして無料で使える.

図 2.2.1: Amazon Echo

他にも Google,Apple,SONY などの大手企業も独自のスマートスピーカー開発し商品化しているが,現段階で Amazon がスマートスピーカーで日本一のシェアを取っている.

2.1.2 Alexa

Alexa とは,2.1.1 節で説明した Amazon Echo に搭載された最新の音声アシスタントのことである.人間の自然な会話でも正しく認識し,操作目的で話しかけた人と周りのノイズ(テレビの音声等)とを正しく分けて認識する .そのため,会話感覚で Alexa と対話することができる.

2.2 マイクロフォンセンサを用いた在宅行動モニタリング

2006 年に行われた実験で,1 ヶ月間マイクロフォンを家の中の床や空間に 6 箇所設置し,在宅内で発生する音圧を計測し行動をモニタリングしたものである.日毎の室音パターンにまとめることで,室音の類似性を比べたところ,平日の行動パターンにある程度規則的な類似性が検出された. [3]

当時の実験では普通の家庭の中で,複数台のマイクロフォンが設置されることは想定されていないと思われる.これからはスマートフォンやスマートスピーカーを含めた複数

台のマイクが室内音をモニタリングすることでユーザーの行動を推測することが可能になると考えられる.

2.3 音状況認識技術（NEC）

NEC が 2016 年に,収集した雑多な音の中から目的の音を切り分け,起きている事象を認識する「音状況認識技術」開発したと発表した.これはあらかじめ目的音をマイクロフォンで収集し,目的音から細かい特徴を構成音として分けて学習させておき,学習していない未知の構成音を環境雑音として分けることで,環境雑音による影響を受けることなく高精度に構成音を抽出するというものである.ガラスの割れた音などを目的音とする事で災害検地や犯罪防止に役立てるという. [4]

異常を検知するために,日常では発生しにくいガラスの割れる音などを目的音として細かく特徴を抽出し,目的音以外の日常で発生する音は環境雑音として処理するという手法をとっている.しかし,この技術をそのまま日常で発生する音を目的音とする今回の提案システムで用いた場合,ユーザーごとの環境音を学習することが必要になることや,正例として学習したい命令を行うよりも前に発生する環境音と,負例として処理したい環境雑音との区別がつかないことが考えられる.

2.4 短時間フーリエ変換(STFT)

音声など時間変化する音響信号の周波数と時間(位相)の解析を行う際に,音響信号全体をフーリエ変換してスペクトルを得てしまうと周波数の時刻毎の変化を知ることが出来ないため,時刻毎のスペクトルを計算してその時間変化を知る処理のことを短時間フーリエ変換(STFT, short-time Fourier transform)という.

短時間フーリエ変換のアルゴリズムは,信号を前から順番に一定の範囲で切り出し,切り出した部分に窓関数をかけてからフーリエ変換を行い,切り出す範囲を少しずらして同じ処理を行うことを繰り返すものである.スペクトル時間変化を表示した図をスペクトログラムと言う.

2.5 畳み込みニューラルネットワーク(CNN)

畳み込みニューラルネットワーク（Convolutional Neural Network, CNN）とは,畳み込みを主な構成要素としたニューラルネットワークである.教師あり学習を前提とした学習を行い,画像認識の分野で非常に高い性能を発揮している.畳み込み層とプーリング層などが積み重なることで多層のネットワークを構築する.

CNN での畳み込み処理は，フィルタと入力画像の内積をとり，ラスタスキャンにより繰り返し畳み込みを行うことで特徴マップを得る．重みフィルタは，誤差逆伝播法による勾配降下最適化法により学習される.

プーリングとは，入力される特徴マップの小領域から値を出力して新たな特徴マップに変換する処理である．プーリングを行う目的は 2 つある．まず，プーリングによりユニット数を減らすため，調整するパラメータを減らすことが出来る．2 つ目は，ある小領域から応答値を出力するため，幾何変化などに対する不変性を獲得することが出来る

人間の手を介さずネットワークの学習を通して画像特徴量の自動抽出が可能になったことで,既存手法を著しく上回る精度を実現した.

2.6 特徴量のスケーリング

特徴量のスケーリングは機械学習の前処理の1つで,標準化と正規化が代表例である.標準化は「平均を 0，分散を 1 とするスケーリング手法」で,正規化は「最小値を 0，最大値を 1 とする 0-1 スケーリング手法」である.こういったスケーリングは,特徴量によって単位が違う場合,または値が極端に違う場合に,各次元の関係をわかりやすくするために有効である.

標準化は英語では"z-score normalization"と呼ばれ,元のデータの平均を 0,標準偏差が 1 のものへと変換する正規化法のことを指す.z-score の求め方は(2.1)式で示す.

$$x_{new}^{i} = \frac{x^{i}-\mu}{\sigma} \tag{2.1}$$

μ は平均を, σ は標準偏差を表す.標準化は説明変数のみに対して行う.

正規化は英語では″min-max normalization″と呼ばれ,データは最大値が 1,最小値が 0 のデータとなるように行われる.正規化の求め方は(2.2)式で示す.

$$x_{new}^{i} = \frac{x^{i} - x_{min}}{x_{max} - x_{min}} \tag{2.2}$$

一般的に標準化を用いる場合は,最大値及び最小値が決まっていない場合や外れ値が存在する場合に利用される.一方,正規化では主に最大値及び最小値が決まっている場合などに利用される.

2.7 クラスタリング

クラスタリングは教師なし学習の 1 種であり,その中でも一般的な学習手法である.クラスタリングとは,ある特徴量空間上のデータをグループ分けし複数のデータのクラスタを作る手法である.クラスタリングは「データをグループ分け」するという説明から頻繁に「分類」と混同されるが,この 2 つは異なった意味の用語である.分類は教師あり学習であり,常にどのグループに所属するかの答えをもとに学習したモデルを使い,答えが未知のデータの所属先を予測する.一方,クラスタリングは教師なしの学習で行うため,データから特徴を学習した上でグループ分けを行う.

クラスタリングを行うためのアルゴリズムには,階層クラスタ分析(＝階層クラスタリング)と非階層クラスタ分析(＝非階層クラスタリング)の 2 種類がある.階層クラスタ分析とは,集合体のデータのうち,最も似ている組み合わせから先にまとめていく階層的手法である.非階層クラスタリングは,階層を作らずにデータをグルーピングしていく手法である.母集団の中で近いデータを収集し,指定された数のクラスタに分類する.この方法では階層クラスタリングとは対照的に,クラスタを形成した後で自由にクラスタを分けることができないため,事前にクラスタ数を指定する必要がある.

2.7.1 k-means クラスタリング

k-means は代表的な非階層的クラスタリング手法である[10][11].以下に k-means の流れを図 2.7.1 に示す.データを点で表し,各クラスタの割り当てを色分けによって表現している.図 2.7.1 より k-means のアルゴリズムは以下のようになる.

1. クラスタ数 k を指定する.

k-means は非階層クラスタリングであるため,事前にクラスタ数を設定する必要がある.

2.ランダムにクラスタを割り当てる.

乱数をもとに,各データにいずれかのクラスタを割り当てる.k-means で得られる結果は乱数による初期値に依存しており,初期値によって局所最適解に陥ることがある.

3. 各クラスタ重心を計算する.

クラスタ毎にデータの重心を求める.重心は「データの平均的な位置」と言い換えることができ,データの平均的な位置を用いて任意の k 個のクラスタに分割するというところから,「k-means」と呼ぶ.

4. クラスタ重心と各データとの距離を計算する.

クラスタ重心と各データのユークリッド距離を計算する.

5. 各データに重心が最も近いクラスタを割り当て直す.

計算した距離から各データに最も距離が近い重心のクラスタに割り当て直す.

6. クラスタの割り当てが収束するまで上記ステップ 3～5 を繰り返す.

割り当て直すことで各クラスタの重心が変化するため,ステップ 3 から 5 を重心が変化しなくなるまで繰り返す.重心が変化しなくなったときクラスタの割り当てが収束したといえる.

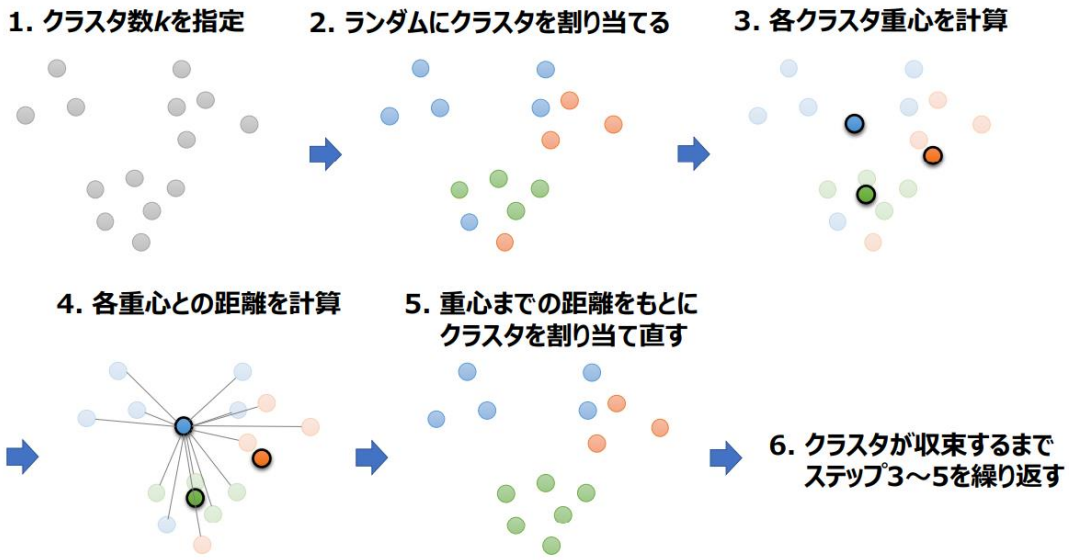

図 2.7.1: k-means のアルゴリズム

代表的なクラスタ数推定手法としてエルボー法があげられる.クラスタ内におけるデータの凝集性に焦点を当て,クラスタ内誤差平方和(SSE)を指標としてクラスタリングの良し悪しを評価する.以下の図 2.7.2 に示すように,クラスタ数を増やしていくと,SSE は減少していく.SSE が 0 であっても,データの数だけクラスタを生成しては意味がない.そこで,クラスタ数に対して SSE をプロットし,SSE とクラスタ数の両者が小さくなるクラスタ数を探る.図 2.7.2 で示すグラフではクラスタ数が 2 から 3 に変化すると SSE は大きく減少しているが,クラスタ数が 3 から 4 に増加してもあまり SSE は減少していない.エルボー法では,このようなひじ(エルボー)のように折れ曲がる点を探す.SSE のプロットから,クラスタ数は 3 つが適切であると考えられる.なお,エルボー法は代表的なクラスタ数推定手法として知られているが,明瞭な折れ曲がりが見られず,適切なクラスタ数の推定が難しい場合もある.

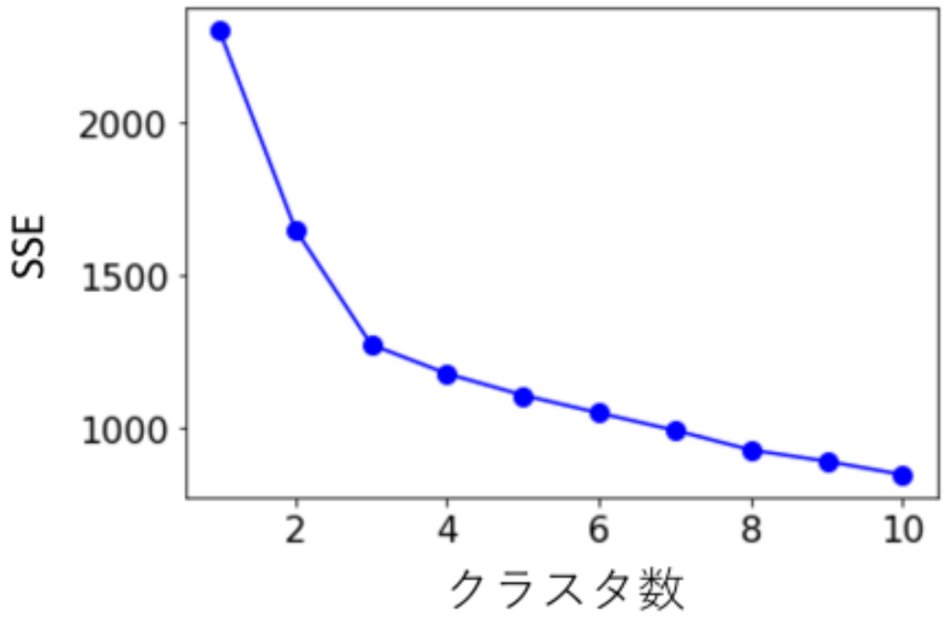

図 2.7.2: エルボー法のグラフ(例)

2.8 主成分分析(PCA)

2.8.1 合成変量の構成

主成分分析(PCA:Principal component analysis) [10] は,広く用いられている線形次元削減手法である.PCA は観測値の分散を維持したまま,観測値をより少ない(ことが期待される)主成分でできた特徴量行列に射影する.PCA は教師なし学習の一つでありターゲットベクトルからの情報を用いず,特徴量行列のみを用いる.

図 2.8.1 に,2 つの変量 x1 と x2 を持つ観測値に主成分分析を行った場合のイメージグラフを示す.図のように,ある程度同質性があると思われる 2 つの変量 x1 と x2 を両軸にとって,対象の集団に属する個々の対象の要素がどのように分布しているかを散布図で描けば,必ずしも 45° の方向とは限らないが,主要な傾向として,左下から右上へ向かう線に沿った分散が見出される.その散布図を仮に楕円と考えた場合,楕円の長経に対して平行な方向である分散が最も大きい方向を第 1 主成分,楕円の長経に対して直角な方向である 2 番目に大きい方向を第 2 主成分と呼び,新たな座標軸をとって,個々の点の座標をこの座標系で表すことができる.

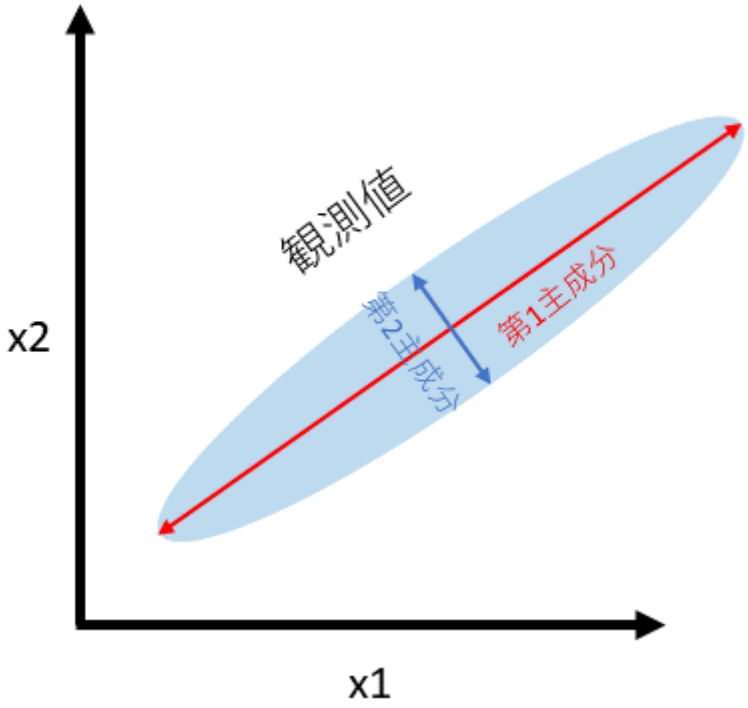

図 2.8.1: 主成分分析(PCA)のイメージグラフ

主成分分析で用いる合成変量とは長径方向の座標や短径方向の座標に相当する数量として構成される.構成の方法は,2 次元での場合,元の変量が x1,x2 であるとき,

$$z = a_1 x_1 + a_2 x_2 \tag{2.3}$$

によって合成変量を定義し,係数について

$$a_1{}^2 + a_2{}^2 = 1 \tag{2.4}$$

という制約を課す.そうすると,この z は, x1 と x2 を成分とするベクトル x の,ある直線の方向への正射影の長さであるといえ,2.4 式が満たされるとき, a_1 と a_2 は何らかの角度の余弦と正弦になる.つまり,

$$a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \tag{2.5}$$

である.したがって,ベクトルa は原点を通って横軸となす角 θ であるような直線 l の方向ベクトルである単位ベクトルとなる.

$$z = a_1 x_1 + a_2 x_2 = \langle a, x \rangle \tag{2.6}$$

は,ベクトル x を直線 l の方向に射影した正射影の長さになる.

2.8.2 合成変量の分散の最大化

以下では対象のデータが n 個である.楕円の長径方向にあたるような z_1 軸,同じことだがベクトル a を,決めるために,元のデータを $x_{1i}, x_{2i}\ (i = 1,2,\cdots\cdots,n)$ として,それらを (2.1) 式にあてはめたとき得られる値を $z_i\ (i = 1,2,\cdots\cdots,n)$ とする.z_i 全体について分散を求め,それが (2.2) 式のもとで最大になるように a_1, a_2を決めると,長径の方向が求まる.

合成変量 z の分散は

$$Var(z) = \frac{1}{n}\sum_{i=1}^{n} (z_i - \bar{z})^2 = {a_1}^2 v_{11} + 2a_1 a_2 v_{12} + {a_2}^2 v_{22} \tag{2.7}$$

となる.(2.7)式の $\bar{z}$ は合成変量 z の平均値 v_{11}, v_{12} は変量 x_1, x_2 の分散であり, v_{12} は変量 x_1, x_2 の共分散である.(2.5) 式ラグランジュ乗数法を用いてラグランジュ関数を構成すると,

$$L(a_1, a_2, \lambda) = {a_1}^2 v_{11} + 2a_1 a_2 v_{12} + {a_2}^2 v_{22} + \lambda(1 - {a_1}^2 - {a_2}^2) \quad (2.8)$$

となる.これを3つの変数について偏微分して整理すると

$$v_{11}a_1 + v_{12}a_2 = \lambda a_1 \quad (2.9)$$

$$v_{21}a_2 + v_{22}a_2 = \lambda a_2 \quad (2.10)$$

$${a_1}^2 + {a_2}^2 = 1 \quad (2.11)$$

ここで (2.9) 式,(2.10) 式は,

$$V = \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}, a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

でおくと,

$$Va = \lambda a \quad (2.12)$$

と書き直せる.行列 V は変量x_1 と x_2 の分散共分散行列とよばれる.$v_{12} = v_{21}$であり

$$Var(z) = \lambda a \quad (2.13)$$

である.(2.12) 式は行列 V の固有値 λ と固有ベクトル a を (2.4) 式の制約のもと求める問題に相当する.先に述べたとおり,分散が最大になるような値を求めればいいの

で,(2.13) 式より,λ の値は 2 つ出てくるので,大きいほうの値をとり, λ_1 とし,それに相当する固有ベクトル a_1 を求める.

このベクトル a_1 によって導きだされる合成変量

$$z_1 = a_{11}x_1 + a_{21}x_2 \tag{2.14}$$

を第1主成分とよぶ.個々の対象のデータをこの式にあてはめたときに得られる z_1 の値

$$z_{1i} = a_{11}x_{1i} + a_{21}x_{2i} \tag{2.15}$$

のことを,対象のデータに関する主成分得点という.

変量が n でも成り立つため,変量が 3 個の場合は,行列 V が 3 × 3 の分散共分散行列になるので,固有値と固有ベクトルが 3 個出る.

2.8.3 寄与率

変量が n 個の場合,固有値 λ と固有ベクトル aは n 個出てくる.ここで

$$\mu_i = \frac{\lambda_i}{\lambda_1+\lambda_2+\cdots+\lambda_n} \tag{2.16}$$

という指標を定義し,第 i 主成分の寄与率とよぶ.μ_iはもとの変量の全変動の何パーセントが第 i 主成分の変動で説明できるか示す指標である.μ_i の値が大きいほどもとの変量の多くを束ねているといえる.さらに,

$$v_i = \sum_{k=1}^{i} \mu_k \tag{2.17}$$

という指標を考え,第 i 番目までの主成分の累積寄与率とよぶ.これは,第1番目から第 i 番目までの主成分によって全体の変動の何パーセントか説明できる.

2.9 既存技術の問題点

アメリカでは,スマートスピーカーを複数台家に設置することで,ユーザーが家のどこにいても,マイクロフォンがユーザーの声を拾い音声操作を可能とするといった,音声操作の普及が進んでいる. 音声操作用のマイクロフォンは音声による命令を受けるために常に周囲の音を収集している.しかし,現状ではスマートフォンをはじめ,スマートスピーカー,IoT 機器といったマイクロフォンを内蔵し音声認識による操作を目的とした入力デバイスは,ユーザーからの命令を AI アシスタントに送ることで動作を行う受動的なものである.これから音声操作可能な機器は増えていくと考えられるが,受動的にしか動かない機器を何台も家に置くだけでは不便な点が多い.ユーザーが発話し AI アシスタントに命令するまで,AI はユーザーの状況を推測することが出来ない.そのため IoT 機器同士がユーザーの状況を共有することが出来ず, 同じサービスを受けるために同じ音声を繰り返すということになる.

また,家庭の中に複数台のマイクが設置されようとしているが,これら複数台のマイクを利用した状況推測システムについて考慮した既存技術は少ない. 既存技術ではガラスが割れる等の異常音を検知するといった緊急の状況を推測することを目的としたものが多いが,日常の動作からユーザーの状況を推測するといった目的のためには,ユーザーごとの環境音を学習することが必要になることや,正例として学習したい命令を行うよりも前に発生する環境音と,負例として処理したい環境雑音との区別がつかないことが考えられる.

第3章 提案システム

3.1 既存技術の問題点の解決

今後,マイクロフォンを使い AI アシスタントとの無線通信が可能な機器を複数台室内に設置することが普及していくと考えられるが,現状の応答システムではユーザーからの命令を待ち続ける受動的な動作しか行わない.ユーザーの状況に合わせたサービスの提案や提供といった能動的な動作を行うには,音声以外の環境音をもとにユーザーの状況を推測することが必要になる.

ユーザーが命令による音声操作を行う時の状況を推測するためには,命令より前の環境音を判別する必要がある.しかしユーザー毎に違う正例となる環境音と,負例となる正例以外の環境音との区別をつけることは容易ではない.ユーザー毎の正例データは時間をかけることで蓄積されていくが,決まった目的音の細かい特徴量をあらかじめ抽出しておく既存技術の手法が行えないために,負例をそれ以外とすることが出来ない.そこで,あらかじめ命令が行われる際の環境音を分析し,数種類のシンボル音を定め分類しておく.ユーザー毎に異なる環境音のなかから,シンボル音をいくつか識別し,シンボル音が発生する順番と時間間隔のパターンを学習することで,命令が行われる状況とパターンとを結びつけることが出来ると考えた.

3.2 提案システムの概要

提案システムについて説明する.あらかじめ,日常的に音声 AI に命令する中で毎日繰り返し命令が行われるような比較的頻度の高い命令を想定し,その時のユーザーの命令を受け付けるよりも前の環境音を分析する.特徴的な環境音を数種類シンボル音として分類しておく.シンボル音の例としては,ドアの鍵が開く音,ドアが開く音,足音,靴を脱ぐ物音などをそれぞれシンボル音とする.

ユーザーの家に設置された複数台の音声 AI が搭載された機器をつかい,命令や一定以上の確度のシンボル音が発生するたびに,命令やシンボル音の種類と発生した時

間とを記録し収集する.収集したデータを用いることで,特定のシンボル音の並びが類似した時間間隔で発生する度に特定の命令が行われているといったパターンを分類することが可能になる.頻度の高い特定の命令の前に発生したシンボル音のパターンとそれに類似したシンボル音の発生パターンを認識したとき,特定の命令を行うことをユーザーに提案する.ユーザーへの提案が承認されることを繰り返す中で,段々と特定の命令を行うかユーザーに提案するのではなく自動的に実行するようになる.

このシステムが完成すれば,『日毎の室音パターンを比べるために録音を保存し続ける』といった既存研究の手法と比べて,記録するのは発生したシンボル音や命令の種類とその時間のデータのみと記録データ量が少なく済む.そして本研究の目的であるユーザーの状況を推測することで,ユーザーに対して命令よりも先にサービスの提案や提供が可能になることでほか,家の中の全ての IoT 機器にユーザーの状況を共有することが可能になるため,家電の管理や制御といった点でも役に立つと考えられる.

3.3 提案システムの課題

提案手法には2つの検証しなければならない課題がある.

1つ目は,①「シンボル音を識別することが可能であるか」の検証である.

2つ目は,②「収集したデータを用いることで,特定のシンボル音の並びが類似した時間間隔で発生する度に特定の命令が行われているといったパターンを分類することが可能であるか」の検証ある.

これら2つの検証によって提案システムが実現可能なものであるかがわかる.

3.3.1 ①への提案手法

命令や一定以上の確度のシンボル音が発生するたびに命令やシンボル音の種類と発生した時間とを記録しデータとして収集するためには,①「シンボル音を識別可能であるかの検証」が必要である.

音声 AI はユーザーからの命令を受ける際,最初にウェイクワードと呼ばれる特定のワード(OK, Google ! や ヘイ Siri !)を認識するまでは命令を受け付けない仕組みになっている.ウェイクワードは音声認識技術を使い複数の音声データを用いてあらかじめ辞書登録することで認識を行う. この仕組みをそのままシンボル音に当てはめることでシンボル音を識別することを考えた場合,環境音であるシンボル音に対してノイズを除去するために複数のフィルタをかける等の方法で精度を高める音声認識手法を当てはめることは困難である.そのため,環境音に短時間フーリエ変換を行い周波数軸と,時間軸によるスペクトログラム画像への変換を行うことで画像データとしてシンボル音の特徴を抽出し,CNN による機械学習を行いシンボル音が識別可能か検証を行う.

3.3.2 ②への提案手法

シンボル音を識別可能であると確認することができたとすると,識別により収集したデータから命令を受け付ける前に発生したシンボル音のパターンを含んだクラスタを見つけ出すことが可能であれば同じクラスタに分類されるような類似した時間間隔で発生したシンボル音のパターンから特定の命令を見つけ出すことが可能になるといえる.

命令やシンボル音の種類と発生した時間とを記録したデータとして検証で使用するデータ列を作成する.作成したデータ列からシンボル音の種類とシンボル音同士の時間間隔を計算したデータ,検証用のラベルとしてシンボル音の発生による順番を含めたデータを作成し,教師なし学習によるクラスタリングを行う.クラスタリングの結果により,シンボル音の並びと時間間隔からパターンを分類することが可能であるか検証を行う.

第4章 実験および結果

4.1 実験概要

本章では,①「シンボル音を識別することが可能であるか」,②「収集したデータを用いることで,特定のシンボル音の並びが類似した時間間隔で発生する度に特定の命令が行われているといったパターンを分類することが可能であるか」検証を行うために2種類の実験を行った.

4.1.1 実験環境

本実験にて使用した実験環境を以下に示す.録音環境はスマートスピーカーの代わりとしてマイクが 2 つ付いた IC レコーダーを使用し,録音したデータの正規化と形式変換は Audacity を,音声データのスペクトログラム画像への変換は Librosa 使用した.音声データは全て wave ファイルで構成されている.

実装は Python にて行い,配列処理等は Numpy, CNN 等のモデル構築と学習は Tensorflow と Keras, クラスタリング等の教師なし学は Scikit-learn,データ分析は Pandas によって行った.

- Ubuntu 16.04 LTS
- NVIDIA GeForce 960M
- CUDA 9.0
- cuDNN 7.0.5
- Audacity
- Librosa
- Python 3
- Numpy 1.15.4
- Tensorflow-gpu 1.12.0
- Keras 2.2.4
- Scikit-learn 0.21.3
- Pandas 0.25.1

4.2 ①に対する実験および結果

環境音の中からそれぞれ異なるシンボル音として識別することが可能であるか検証を行うために,人間でも異なるシンボル音同士の識別が行えているかが判りやすい音を想定して選び出し,機械学習を行い,識別が正しく行えているかの検証を行った.

4.2.1 実験方法

人間でも識別が正しく行われているかを判別できる特徴音として, ユーザーが帰宅した際に音声 AI に照明をつけるように命令する状況を想定し,外側から鍵を開ける音,ドアの開閉音,内側から鍵を閉める音の 3 つの特徴音を選んだ.

3 つの特徴音をそれぞれ 120 個ずつ録音したデータを用意した.録音方法は IC レコーダーを玄関の扉から 1.2 メートル離れた位置に高さ 1.0 メートル程の高さで設置し,シンボル音候補となる 3 つの特徴音をそれぞれ数秒間ずつ収録した.

録音したシンボル音をスペクトログラムになるように短時間フーリエ変換を行った.スペクトログラムにおいて,横軸が時間,縦軸が周波数,色が白いほど音圧が強いことを示している.これを CNN の学習に利用し易くするためにスペクトログラムを 380 ＊ 380 pix の画像に変換し CNN の入力画像とした. 入力画像の例の一部を以下の図 4.2.1,図 4.2.2,図 4.2.3,で示す. 図 4.2.1 の画像は” 外側から鍵の開く音”のスペクトログラムの画像,図 4.2.2 は ” ドアの開閉音”のスペクトログラムの画像,図 4.2.3 の”内側から鍵の閉める音”のスペクトログラム画像である.

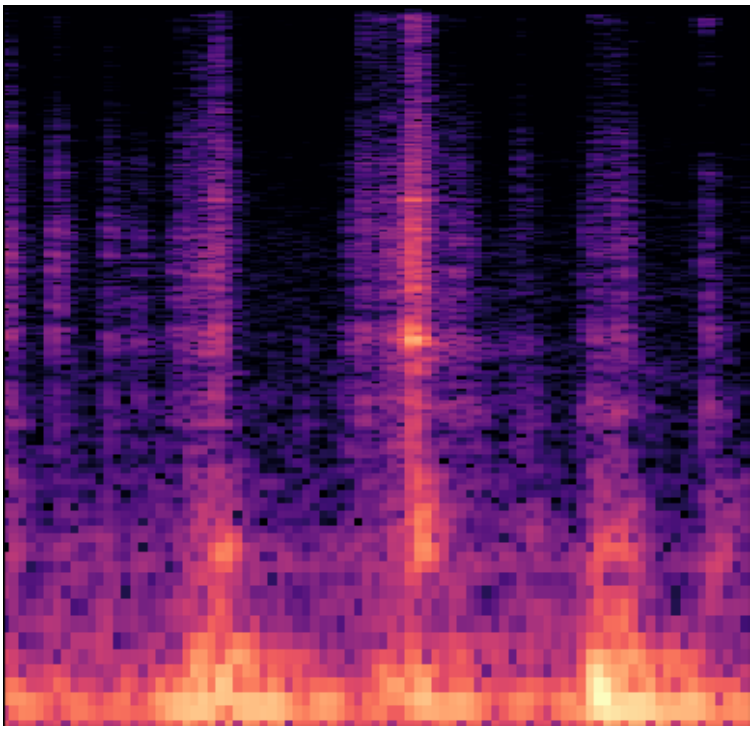

図 4.2.1: ” 外側から鍵の開く音”のスペクトログラム画像

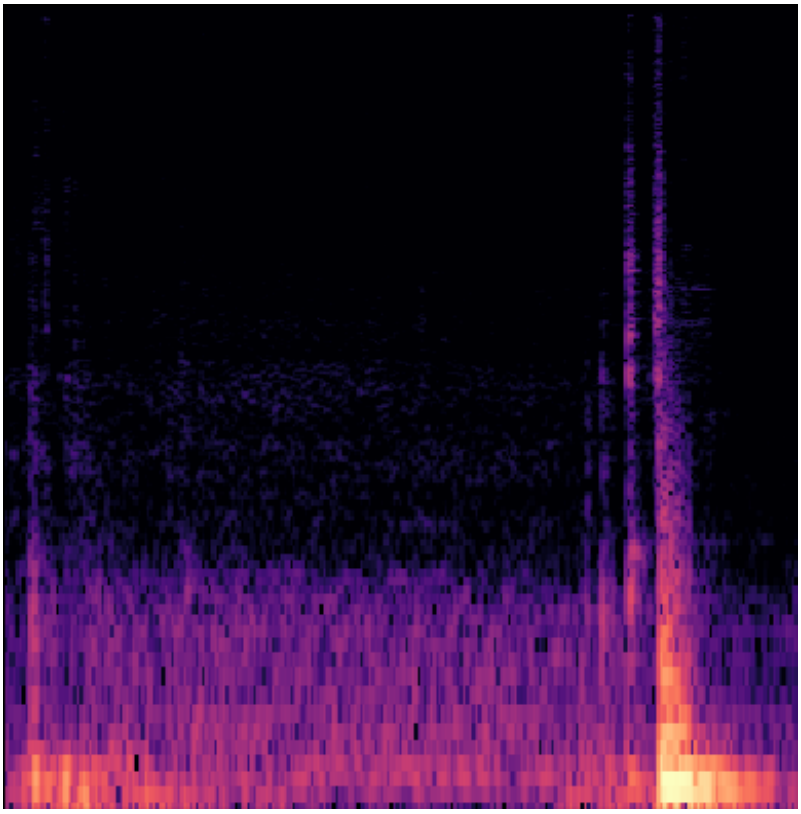

図 4.2.2: ” ドアの開閉音”のスペクトログラム画像

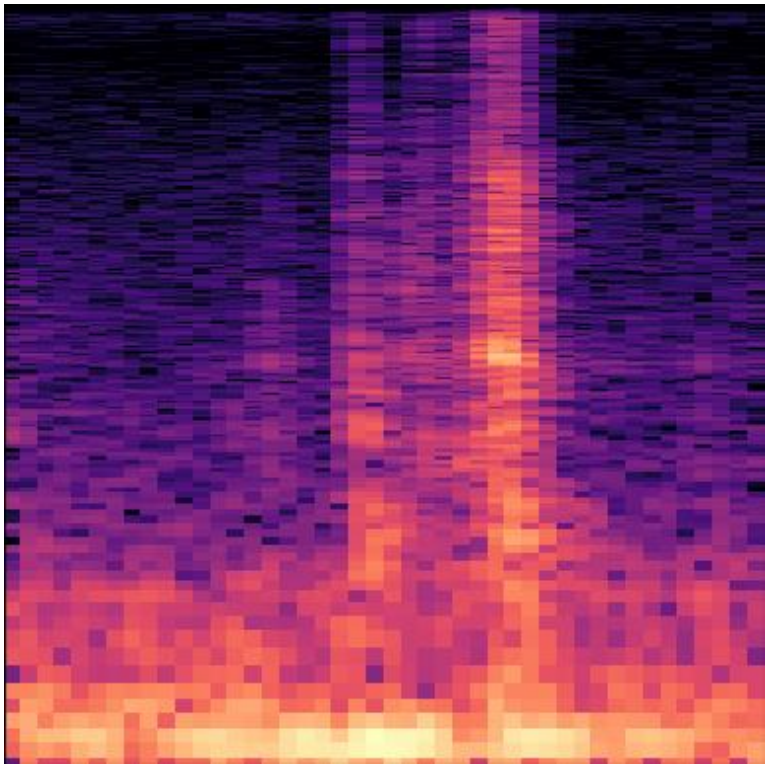

図 4.2.3: ”内側から鍵の閉める音”のスペクトログラム画像

4.2.2 実験結果

鍵の開ける音,ドアの開閉音,鍵の閉める音を画像データ毎にラベリングし,それぞれ 120 枚のうち 80 枚を学習用データに 20 枚を検証用データにして CNN モデルを構築し学習を行った. CNN モデルの学習結果として識別の精度と損失値をグラフにし図 4.2.4, 図 4.2.5 に示す.図 4.2.4 の縦軸はモデルの識別精度,図 4.2.5 の縦軸はモデルの損失値,横軸は epoch(データセット全体の処理の単位,その反復回数),点線グラフは学習用データ,折れ線は検証用データの結果である. 図 4.2.5, 図 4.2.6 のグラフから epoch が進むごとに識別精度,損失値ともに学習用データと検証用のデータの差が開いているため,画像データが少なく過学習の傾向が見られる.

学習の済んだモデルで未確認データ各 20 個ずつの識別を行いその予測精度から識別が可能か検証した. 未確認データを識別したときのモデルの予測精度の損失値は

0.350936,識別精度は 0.978125 となった. CNN モデルの識別精度が未確認データに対して平均 97.8%と高い精度で識別が可能であることがわかった

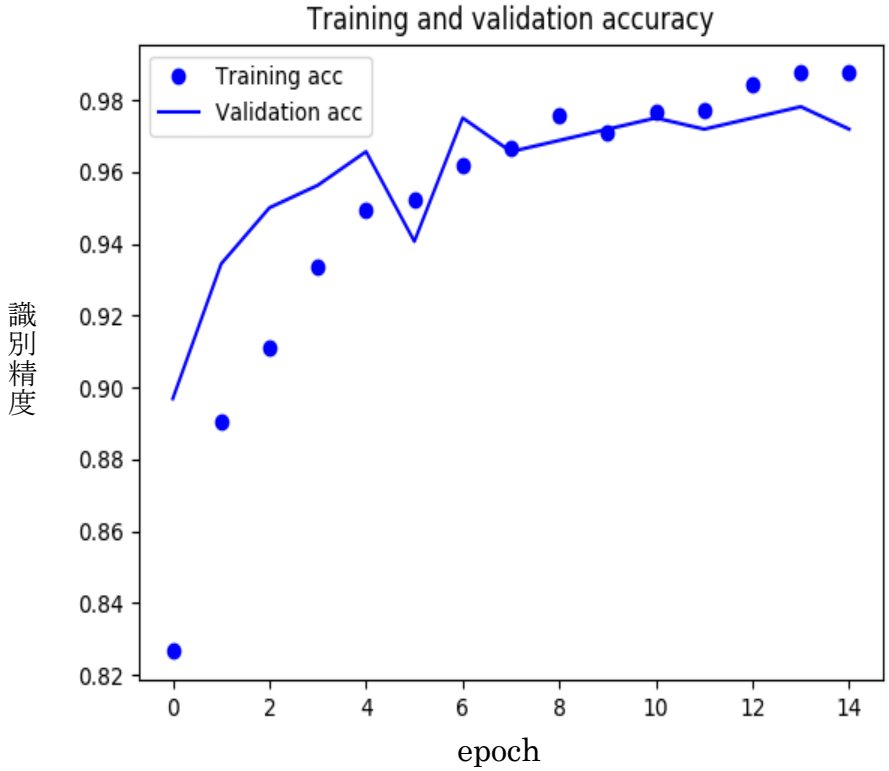

図 4.2.4: 精度を表すグラフ

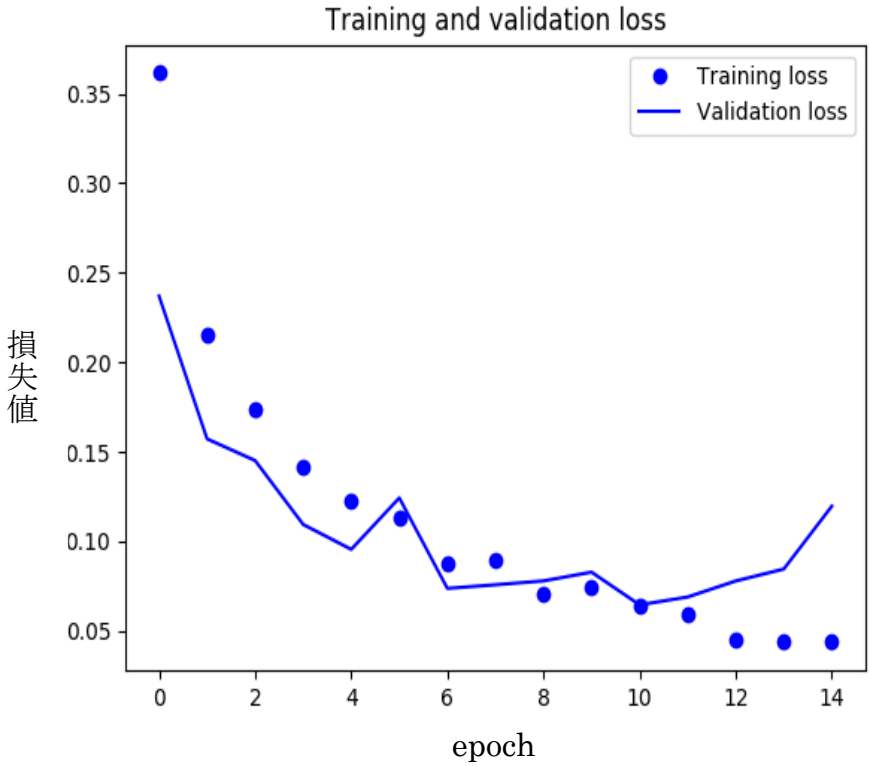

図 4.2.5: 損失値を表すグラフ

4.3 ②に対する実験および結果

4.3.1 データ

本実験では,シンボル音の並びと時間間隔からパターンを分類することが可能であるか検証を行うために,3 種類のシンボル音が識別されたものとして実際に録音した音を基に検証用のダミーデータを作成した.

①の実験と同様に,提案システムが実際に使用される状況として繰り返し同じサービスを受けるために同じ音声操作をユーザーが行う状況として,ユーザーが帰宅した際に照明をつけるために音声操作を行う状況を想定した.その時に発生したシンボル音として,ドアの鍵を開ける音,ドアが開閉する音,鍵を閉める音の 3 種類を実際に録音し,その種類と発生した時間を記録したデータ列を作成した.以下の表 4.3.1 に記録したデータ列を示す.表 4.3.1 に示した通りに,10 個の録音データからA:ドアの鍵を開ける音,B:ドアが開閉する音,C:鍵を閉める音について発生した順番と,発生した順番毎の発生時間についてそれぞれデータとして記録し,検証用に発生順を基にしたパターンを正解ラベルとして列に加えた.

表 4.3.1: 録音したA, B, Cの順番と発生時間のデータ列

ID	A	B	C	1(s)	2(s)	3(s)	label
1	1	2	3	2.25201	7.57904	10.58902	abc
2	1	2	3	13.67685	19.60711	20.881	abc
3	1	2	3	10.91042	13.87936	17.11763	abc
4	1	2	3	6.685157	12.58007	14.79554	abc
5	1	2	3	9.856409	16.06344	18.28378	abc
6	1	2	3	12.98756	18.97865	25.73841	abc
7	1	2	3	10.54514	14.7109	22.98938	abc
8	1	2	3	7.351212	11.18272	17.65112	abc
9	1	2	3	9.473455	11.22134	16.20159	abc
10	1	2	3	12.09237	15.85485	16.94273	abc

以下の表 4.3.2 にダミーデータ列を加えたデータ列の表を示す. 表 4.3.2 に示した通りに,録音した発生時間を基にランダムに設定した発生時間を持つダミーデータを50個

用意し,ACB, BAC, BCA,CAB, CBAの5種類の発生順に10個ずつ振り分けたデータ列を用意し,表 4.3.1 のデータ列にダミーデータを加えた計 60 個のデータ列を作成した.

表 4.3.2: 録音データにダミーデータを加えたデータ列(適宜略)

ID	A	B	C	1(s)	2(s)	3(s)	label
1	1	2	3	2.25201	7.57904	10.58902	abc
2	1	2	3	13.67685	19.60711	20.881	abc
3	1	2	3	10.91042	13.87936	17.11763	abc
4	1	2	3	6.685157	12.58007	14.79554	abc
16	1	3	2	9.066543	11.8	20.51449	acb
17	1	3	2	8.721658	11.60403	20.38556	acb
18	1	3	2	11.91862	16.18877	18.99218	acb
19	1	3	2	12.84164	21.00573	26.30923	acb
20	1	3	2	9.95237	15.25752	19.16762	acb
21	2	1	3	7.698212	15.30699	19.56896	bac
22	2	1	3	10.81737	13.33155	15.0579	bac
23	2	1	3	13.28968	16.10108	24.53068	bac
24	2	1	3	11.07606	14.29509	23.76503	bac
25	2	1	3	10.08003	14.73367	20.63285	bac
51	2	3	1	11.90575	16.32	22.16911	cab
52	2	3	1	6.316384	14.37961	17.01015	cab
53	2	3	1	11.81119	17.87949	25.22871	cab
54	2	3	1	12.68048	21.806	26.22691	cab
55	2	3	1	6.243388	12.44603	13.45839	cab
56	2	3	1	10.10231	13.37852	19.95493	cab
57	2	3	1	8.184177	11.70028	18.14884	cab
58	2	3	1	8.405945	13.01298	16.86776	cab
59	2	3	1	10.14545	19.14601	24.81389	cab
60	2	3	1	12.28418	18.67012	21.86744	cab

以下の表 4.3.3 に作成した発生時間から発生間隔を計算したデータ列の一部を示す. 表 4.3.2 のデータ列の項目である3つの発生した時間「1(s)」,「2(s)」,「3(s)」から,表

4.3.3 に示す通りに発生した時間の間隔を計算したデータ列の項目として「1(s)」と「2(s)」の時間の間隔を「1to2」,「2(s)」と「3(s)」の時間の間隔を「2to3」として変換を行った.

表 4.3.3: 発生時間から発生間隔を計算したデータ列(一部)

ID	A	B	C	1to2	2to3	label
1	1	2	3	5.32703	3.00998	abc
2	1	2	3	5.930257	1.273898	abc
3	1	2	3	2.968943	3.238267	abc
4	1	2	3	5.89491	2.215471	abc
5	1	2	3	6.207032	2.220335	abc
6	1	2	3	5.991093	6.759756	abc
7	1	2	3	4.165765	8.27847	abc
8	1	2	3	3.831509	6.468401	abc
9	1	2	3	1.747884	4.98025	abc
10	1	2	3	3.762487	1.087878	abc
11	1	3	2	2.80708	5.102513	acb
12	1	3	2	1.304542	6.592312	acb
13	1	3	2	6.289866	6.734536	acb
14	1	3	2	3.580978	2.727961	acb
15	1	3	2	3.991325	5.278341	acb
16	1	3	2	2.821652	8.626298	acb
17	1	3	2	2.882371	8.781527	acb
18	1	3	2	4.270155	2.803404	acb
19	1	3	2	8.164099	5.303492	acb
20	1	3	2	5.305149	3.910104	acb
21	2	1	3	7.608777	4.261971	bac
22	2	1	3	2.514177	1.726357	bac
23	2	1	3	2.811401	8.429597	bac
24	2	1	3	3.21903	9.469946	bac
25	2	1	3	4.653641	5.89918	bac
26	2	1	3	9.458044	6.837707	bac
27	2	1	3	2.048076	6.802284	bac
28	2	1	3	2.538554	5.3849	bac
29	2	1	3	5.488643	1.431481	bac
30	2	1	3	6.095814	2.335884	bac

次に ID,と label 以外の項目の標準化処理を行った.以下の表 4.3.4 にそのデータの一部を示す. 表 4.3.4 が示すとおりにクラスタリングを行う前の下準備としてデータのスケーリングが行えたことがわかる.

表 4.3.4: 標準化によりスケーリングしたデータ列(一部)

ID	A	B	C	1to2	2to3	label
1	-1.22474	0	1.224745	0.008585	-0.93508	abc
2	-1.22474	0	1.224745	0.278321	-1.71048	abc
3	-1.22474	0	1.224745	-1.04578	-0.83313	abc
4	-1.22474	0	1.224745	0.262516	-1.28994	abc
5	-1.22474	0	1.224745	0.402076	-1.28777	abc
6	-1.22474	0	1.224745	0.305523	0.739676	abc
7	-1.22474	0	1.224745	-0.51064	1.41798	abc
8	-1.22474	0	1.224745	-0.6601	0.609548	abc
9	-1.22474	0	1.224745	-1.59176	-0.05511	abc
10	-1.22474	0	1.224745	-0.69096	-1.79356	abc
11	-1.22474	1.224745	0	-1.11815	-0.0005	acb
12	-1.22474	1.224745	0	-1.78999	0.66489	acb
13	-1.22474	1.224745	0	0.439114	0.728412	acb
14	-1.22474	1.224745	0	-0.77212	-1.06105	acb
15	-1.22474	1.224745	0	-0.58864	0.078031	acb
16	-1.22474	1.224745	0	-1.11164	1.573331	acb
17	-1.22474	1.224745	0	-1.08449	1.642661	acb
18	-1.22474	1.224745	0	-0.46397	-1.02735	acb
19	-1.22474	1.224745	0	1.277146	0.089264	acb
20	-1.22474	1.224745	0	-0.00119	-0.53307	acb
21	0	-1.22474	1.224745	1.028843	-0.37591	bac
22	0	-1.22474	1.224745	-1.24912	-1.50839	bac
23	0	-1.22474	1.224745	-1.11622	1.485478	bac
24	0	-1.22474	1.224745	-0.93396	1.95013	bac
25	0	-1.22474	1.224745	-0.2925	0.355316	bac
26	0	-1.22474	1.224745	1.855711	0.774491	bac
27	0	-1.22474	1.224745	-1.45753	0.75867	bac
28	0	-1.22474	1.224745	-1.23822	0.125623	bac
29	0	-1.22474	1.224745	0.080861	-1.6401	bac
30	0	-1.22474	1.224745	0.352347	-1.23616	bac

4.3.2 k-means クラスタリング

表 4.3.4 に示す作成した標準化されたデータ列をもとに発生順の項目'A', 'B', 'C'と発生時間間隔の項目'1to2', '2to3'のデータを利用して k-means によるクラスタリングを行った.k-means では予めクラスタ数を決める必要があるため,クラスタ数推定手法としてエルボー法を用いた. 実験で用いたエルボー法のグラフを以下の図 4.3.1 に示す.横軸をクラスタ数 K,縦軸をクラスタ内誤差平方和(SSE)とするグラフとして作成を行った.

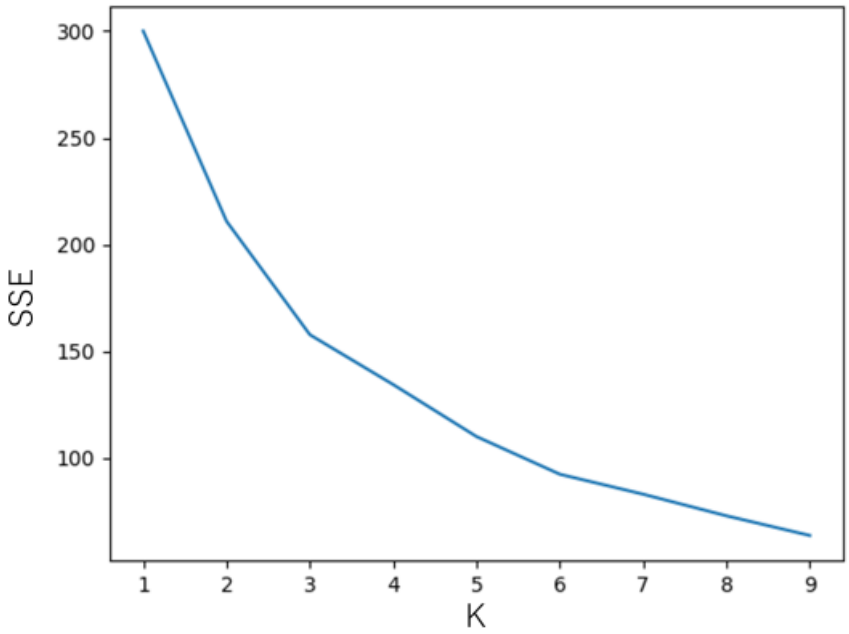

図 4.3.1: 実験で使用したエルボー法のグラフ

図 4.3.1 のエルボー法のグラフからクラスタ数が 6 から 7 に増加するときが一番グラフの変動が少なくなっていることがわかるため,予め設定するクラスタの数は 6 に決めた.k-means によるクラスタリングを行った結果が色分けされた散布図行列で表したグラフを以下の図 4.3.2 で示す.図 4.3.2 の散布図行列は発生順の項目'A', 'B', 'C'と発生時間間隔の項目'1to2', '2to3'の 5 つの項目から全ての 2 項目同士の組み合わせに対して散布図を作成し,同一の項目となる対角線上には項目毎のヒストグラムを表示した.図 4.3.2 の散布図行列で可視化し確認した限りでシンボル音の発生順に結び付くものは見つからなかった.

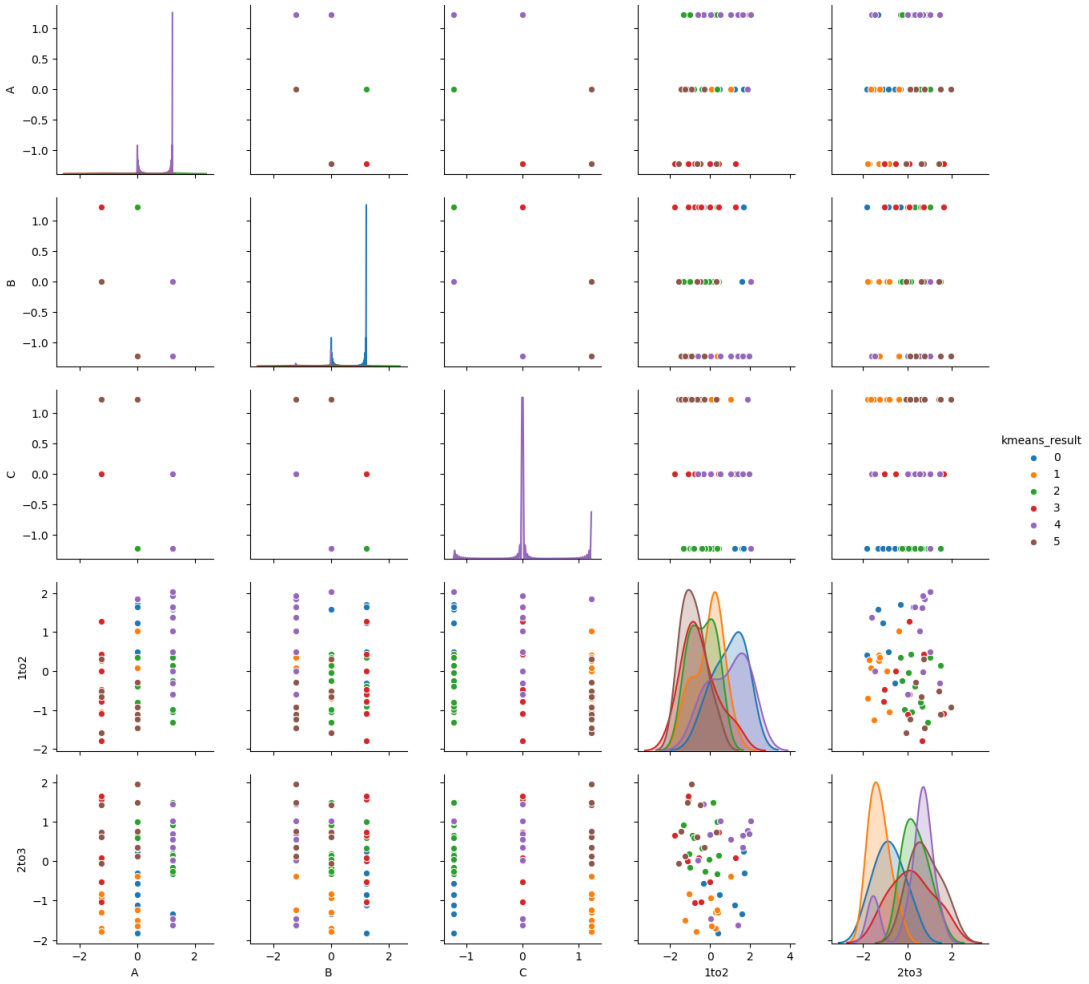

図 4.3.2: 5 次元の k-means クラスタリングで色分けされた散布図行列

4. 3. 3 主成分分析(PCA)による次元削減

表 4.3.4 に示す作成した標準化されたデータ列をもとに発生順の項目'A', 'B', 'C'と発生時間間隔の項目'1to2', '2to3'の 5 項目のデータを PCA によって次元削減を行い, 第 1 主成分と第 2 主成分の 2 次元の合成変数として出力した.PCA で出力した 2 次元の合成変数のグラフを以下の図 4.3.3 に示す.横軸を第 1 主成分(pca1),縦軸を第 2 主成分(pca2)とする散布図としてグラフを作成した.比較するために, 図 4.3.3 の散布図を'label'項目で色分けした散布図を以下の図 4.3.4 に示す.また,出力された第 1 主成分と第 2 主成分の寄与率を以下の表 4.3.5 に示す.

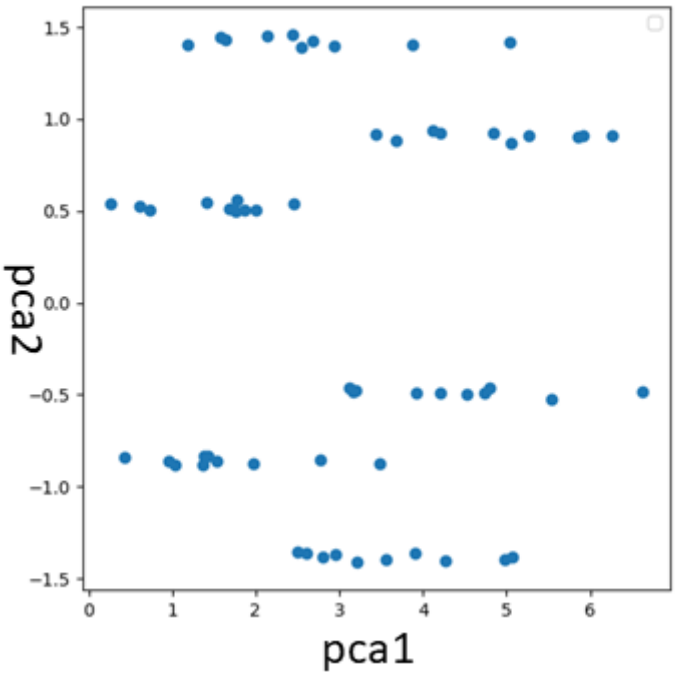

図 4.3.3:主成分分析(PCA)で出力した 2 次元の合成変数の散布図

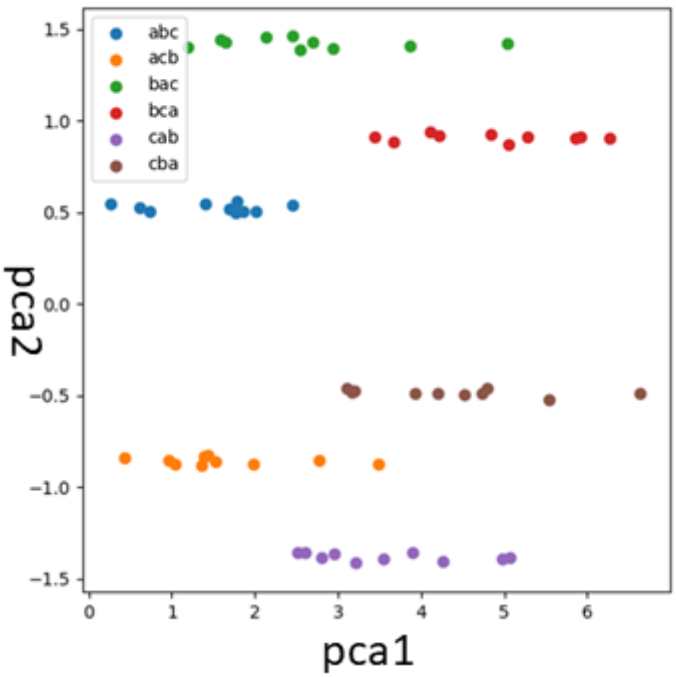

図 4.3.4: 正解ラベルにより色付けられた図 4.3.3 の散布図

表 4.3.5: 第 1 主成分と第 2 主成分の寄与率

pca1	pca2
0.34933541	0.30000551

図 4.3.4 を見ると同一ラベルのデータが層状にグループしていることがわかる.表 4.3.5 から第 1 主成分,第 2 主成分ともに 30%以上の寄与率を持つことがわかる.

4.3.4 主成分分析で次元削減後に k-means クラスタリング

PCA によって出力された第 1 主成分と第 2 主成分の 2 つの合成変数のデータ列を使い,2 つの k-means の学習を行った.第 1 主成分のデータのみで学習した k-means によるクラスタリング結果を以下の図 4.3.5 に,第 2 主成分のデータのみで学習した k-means によるクラスタリング結果を以下の図 4.3.6 に示す. 図 4.3.3 と同様に,横軸を第 1 主成分 (pca1) ,縦軸を第 2 主成分(pca2)の散布図としてグラフを作成し,図 4.3.5 では第 1 主成分のデータのみで学習した k-means によるクラスタリング結果による色分け,図 4.3.6 では第 2 主成分のデータのみで学習した k-means によるクラスタリング結果による色分けを行った.

図 4.3.5 を見ると,第 1 主成分によってシンボル音の発生間隔の類似度がわかる. 図 4.3.6 を見ると第2主成分によって特定のシンボル音の発生順に分けることができることがわかる.

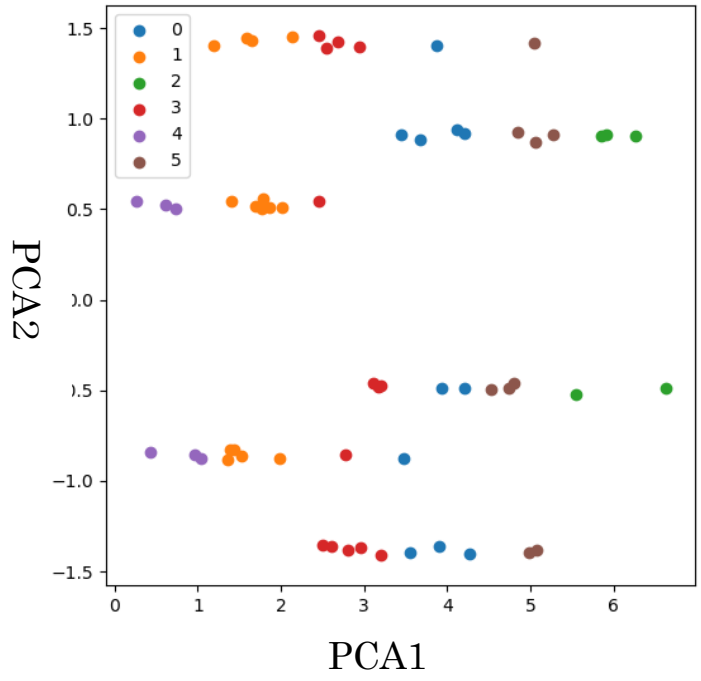

図 4.3.5: 第 1 主成分のデータを学習した k-means の結果

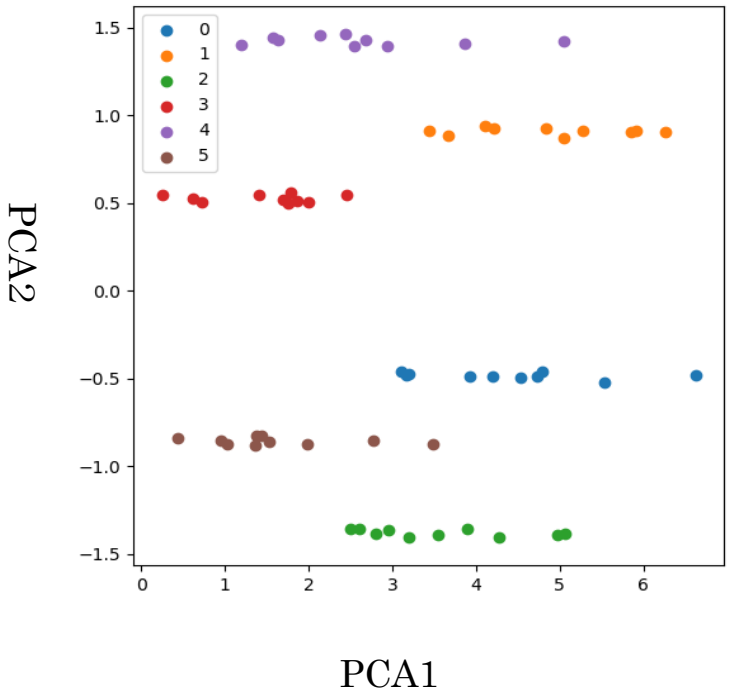

図 4.3.6: 第 2 主成分のデータを学習した k-means の結果

第5章 考察

5.1 実験結果１の考察

1 つのシンボル音につき学習用データ 80 個,検証用データ 20 個のスペクトログラム画像でCNNの学習を行った結果,CNN モデルの識別精度が未確認データに対して平均 97.8%と高い精度で識別が可能であることがわかった.図 4.2.4, 図 4.2.5 のグラフから検証用のデータは画像データが少なく過学習が起こっていることがわかるが,図 4.2.7 の未確認データに対する識別精度の高さから 80 枚で行った学習で十分に学習が進んでいたことがわかった.

個別のシンボル音を個別のものとして識別が可能であることがわかったため, 命令や一定以上の確度のシンボル音が発生するたびに命令やシンボル音の種類と発生した時間とを記録しデータとして収集することが可能である.また,収集したデータをテキストデータの列として扱うことで,パターン分類等の学習用のデータとして使用する際に画像や音響データと比べて軽いものになった.

5.2 実験結果２の考察

5 つの項目を学習させた 5 次元の k-means によって 6 つのクラスタに分類を行ったが,図 4.3.2 の散布図行列で可視化し確認した限りでシンボル音の発生順に結び付くものは見つからなかった.これは次元数が多いことが原因と考えられるため,主成分分析による次元削減を行う.

主成分分析による次元削減を行い,第 1 主成分と第 2 主成分の 2 つの合成変数を出力した.表 4.3.5の寄与率を見ると第 1 主成分と第 2 主成分には元データから約65%の情報が反映されていることが示されている.主成分分析では第 2 主成分までの累計寄与率が 50%を下回るとデータとして使えないと判断されるが,今回の実験結果では2つの主成分が 30%以上の寄与率を持っているためデータとして扱えると考えられる.第 1 主成分と第 2 主成分を散布図にすると図 4.3.3が示すようにクラスタが6つに分

かれることが確認された.正解ラベルによって色付けしたところ図 4.3.4 が示すようにシンボル音の発生順で6つのグループ分けされていることがわかる.このことから,第 1 主成分には発生時間の間隔における項目が関係しており,第 2 主成分には発生順の項目が関係していると考えられる.

PCA によって出力された第 1 主成分と第 2 主成分の 2 つの合成変数のデータ列を使い,2 つの k-means の学習を行った結果の図 4.3.5,図 4.3.6 を見ると,第 1 主成分によってシンボル音の発生間隔の類似度がわかり,第 2 主成分によって特定のシンボル音の発生順に分けることができることがわかる.

これにより,第 1 主成分から大きく時間間隔が離れているシンボル音の排除が可能になり,第 2 主成分から未知のデータに対しても k-means がクラスタ重心とデータの距離からクラスタを分類する特性によって分類することができるといえる.

これにより,識別により収集したデータから命令が行われる前に発生したシンボル音の時間的なパターンと順序的なパターンの分析が可能であり同一のパターンを含んだクラスタを見つけ出すことで同じクラスタに分類されるような類似したシンボル音のパターンから特定の命令を見つけ出すことが可能になるといえる.

5.3 全体の考察

シンボル音の識別においては,個別のシンボル音を個別のものとして識別が可能であることが検証されたため, 命令や一定以上の確度のシンボル音が発生するたびに命令やシンボル音の種類と発生した時間とを記録しデータとして収集することが可能であるとわかった.識別により収集したデータから命令を受け付ける前に発生したシンボル音のパターンを含んだクラスタを見つけ出すことが可能であることがわかった.そのため同じクラスタに分類されるような類似した時間間隔で発生したシンボル音のパターンから特定の命令を見つけ出すことが可能になるといえるようになった.

これら2つの検証によって提案システムが実現可能なものであるとわかった.

第6章 おわりに

本研究では,マイクロフォンを内蔵し音声認識による操作を目的とした入力デバイスを利用し環境音を用いて人の動作を推定するシステムの提案, システムの実現に必要な検証として①特定の環境音の識別が可能であることを検証し,②識別により収集したデータから特定のシンボル音のパターンが分類可能であることを検証した.

今後の展望としましては, ①, ②ともに人間が識別できる環境音での実験であったため, システムの実現性を高めるためにシンボル音になり得る機械だけが認識するような環境音に対しても実験を繰り返す必要が挙げられる.

謝辞

本論文を作成するにあたり,多くのご指導,ご助言を頂きました三好力教授に厚くお礼を申し上げます.また,議論に協力してくださった三好研究所の皆様や学友に心から感謝致します.

参考文献

[1] 野澤哲生, AI スピーカー日本上陸,API が家電を支配する, 日経エレクトロニクス, https://tech.nikkeibp.co.jp/dm/atcl/mag/15/00176/00001 (参照 2017-10-19).

[2] 猿渡俊介, “IoT の原点,センサーネットワークの本質” , 日経エレクトロニクス,https://tech.nikkeibp.co.jp/dm/atcl/mag/15/100600071 (参照 2015-12-11).

[3] 川佳満, 岸本 俊夫, 太田 茂, “マイクロフォンセンサを用いた在宅行動モニタリング”,川崎医療福祉学会誌 Vol.15 No.2 2006 615-620 (2005).

[4] 日本電気株式会社, “NEC 音状況認識技術” 日本電気株式会社 https://jpn.nec.com/press/201611/20161128_03.html (参照 2017-10-19).

[5] 青木茂樹, 岩井嘉男, 大西正輝, 小島篤博, 福永邦雄, ”人物の位置・姿勢に注目した行動パターンの学習・認識と非日常状態検出への応用”, 電子情報通信学会論文誌, Vol. J87-D- II No.5 (2004-4).

[5] 林知樹, 戸田智基, “統計的手法による音響イベント検出”, 音響学会誌, 75, 532-537 (2019).

[6] T. Virtanen, M. D. Plumbley and D. Ellis, “Introduction to sound scene and event analysis”, in Computational Analysis of Sound Scenes and Events (2017), pp. 303-333.

[7] 井本桂右, “音響イベントと音響シーンの分析” 音響学会誌, 74, 198-207 (2018).

[8] 井本桂右, “環境音分析の研究動向”, 音響学会誌, 75, 512-518 (2019).

[9] 佐々木逸士, 佐々木健, “非音声の環境音認識に関する研究”, 精密工学会学術講演会講演論文集, 2015 年度精密工学会春季大会, Q38, (2015-03-01).

[10] Chris Albon,中田秀基訳, “Python 機械学習クックブック”, オライリージャパン (2018).

[11] システム計画研究所, “Python による機械学習入門”, オーム社 (2016).

発表履歴

[1]北迫祐樹："AI スピーカーを利用した状況推測システムの提案," 2018 年度龍谷大学卒業論文, t150477 (2019)

[2]KITASAKO Yuki, MIYOSHI Tsutomu："Proposal of situation estimation system using AI speaker," Proceedings of the 11th International Conference on Information Science and Application 2020 (ICISA2020), Virtual Conference, 35671 (2020).

[3]Yuki KITASAKO, Tsutomu MIYOSHI："Proposal of Situation Estimation System Using AI Speaker," Lecture Notes in Electrical Engineering 739 Information Science and Applications Proceedings of ICISA2020, Springer, ISBN:978-981-33-6384-7, ISBN:978-981-33-6385-4, pp.287-291 (2021).

付録

Spectrogram.py

```
# -*- coding: utf-8 -*-

from __future__ import print_function
import librosa
import librosa.display
import numpy as np
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt

f = ['1_1_1.wav']
p = ['1_1_1.png']
for (wavename, pingname) in zip(f, p):
# ファイル読込
                audio_path = wavename
                y, sample_rate = librosa.load(audio_path)

# 短時間フーリエ変換
                S = np.abs(librosa.stft(y))

# 画像出力
                plt.figure(figsize=(6,4))
                librosa.display.specshow(librosa.amplitude_to_db(S, ref=np.max), y_axis='log', x_axis='time')
                plt.title('Power spectrogram')
                plt.colorbar(format='%+2.0f dB')

                plt.tight_layout()
# save as png
                plt.savefig(pingname)
```

Make_data.py

```
# -*- coding: utf-8 -*-

from sklearn.model_selection import train_test_split
from PIL import Image
import os,glob
import numpy as np
import shutil
import math
import random

#分類のカテゴリーを選ぶ
root_dir = 'image/spectrogram_'
train_dir = 'image/train/'
test_dir = 'image/test/'
groups = ['door','noise']
nb_classes = len(groups)
image_size = 128
print('----door を処理中----')
image_dir = root_dir + 'door'
move_train_dir = train_dir + 'door'
move_test_dir = test_dir + 'door'
#files = glob.glob(image_dir+'/*.png')
print(len(files))
th = math.floor(len(files)*0.2)
random.shuffle(files)
#データの 20%を test ディレクトリに移動させる
for i in range(660):
   shutil.move(files[i],move_test_dir)
#残りすべてを train ディレクトリに移動させる
files = glob.glob(image_dir+'/*.png')
for file in files:
   shutil.move(file,move_train_dir)

print('----noise を処理中----')
image_dir = root_dir + 'noise'
move_train_dir = train_dir + 'noise'
move_test_dir = test_dir + 'noise'
files = glob.glob(image_dir+'/*.png')
print(len(files))
th = math.floor(len(files)*0.2)
random.shuffle(files)
#データの 20%を test ディレクトリに移動させる
for i in range(242):
```

```
  shutil.move(files[i],move_test_dir)
#残りすべてを train ディレクトリに移動させる
files = glob.glob(image_dir+'/*.png')
for file in files:
  shutil.move(file,move_train_dir)
"""画像データを Numpy 形式に変換"""
root_dir = 'image/train/'
#訓練データ
#フォルダごとの画像データを読み込む
X = []
Y = []

for idx,group in enumerate(groups):
    image_dir = root_dir + group
    files = glob.glob(image_dir+'/*.png')
    print('----{}を処理中----'.format(group))
    for i,f in enumerate(files):
        img = Image.open(f)
        img = img.convert('RGB') #カラーモードの変更
        img = img.resize((image_size,image_size))#画像サイズの変更
        data = np.asarray(img)
        X.append(data)#画像をベクトルにしたもの
        Y.append(idx)#二値化問題
X = np.array(X)
Y = np.array(Y)

#学習用データと検証用データに分ける
X_train,X_test,y_train,y_test = train_test_split(X,Y,random_state=0)
xy = (X_train,X_test,y_train,y_test)
np.save('data/train.npy',xy)
print(X_train.shape[1:])
print('ok',len(Y))
root_dir = 'image/test/'
#テストデータ
#フォルダごとの画像データを読み込む
X = []
Y = []
for idx,group in enumerate(groups):
    image_dir = root_dir + group
    files = glob.glob(image_dir+'/*.png')
    print('----{}を処理中----'.format(group))
    for i,f in enumerate(files):
        img = Image.open(f)
        img = img.convert('RGB') #カラーモードの変更
        img = img.resize((image_size,image_size))#画像サイズの変更
        data = np.asarray(img)
        X.append(data)#画像をベクトルにしたもの
        Y.append(idx)#二値化問題
X = np.array(X)
Y = np.array(Y)

#テストデータ保存
np.save('data/X_test.npy',X)
np.save('data/y_test.npy',Y)
print(X_train.shape[1:])
print('ok',len(Y))
```

CNN.py

```
# -*- coding: utf-8 -*-

import numpy as np
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D
from keras.layers import Activation,Dropout,Flatten,Dense
from keras.utils import np_utils
import matplotlib.pyplot as plt
#カテゴリ
groups = ['door','noise']
nb_classes = len(groups)

#モデル構築
def build_model(in_shape):
    model = Sequential()
    model.add(Conv2D(32,(3,3),padding='same',input_shape=in_shape))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2),data_format='channels_first'))

    model.add(Conv2D(64,(3,3),padding='same'))
    model.add(Activation('relu'))
    #model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(128,(3,3)))
    model.add(Activation('relu'))
```

```
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Conv2D(128,(3,3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Flatten())
    model.add(Dropout(0.5))

    model.add(Dense(512))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    model.add(Dense(nb_classes))
    model.add(Activation('sigmoid'))

#モデル構成の確認
    #model.summary()
    model.compile(loss='binary_crossentropy',optimizer='rmsprop',metrics=['accuracy'])
    return model

#正解率,損失値のグラフ
def draw_graph(model):
    fig, (axL, axR) = plt.subplots(ncols=2, figsize=(16,8))

    acc = model.history['acc'] #学習用データの正解率
    val_acc = model.history['val_acc'] #検証用データの正解率
    loss = model.history['loss'] #学習用データの損失値
    val_loss = model.history['val_loss'] #検証用データの損失値

    axL.plot(acc,label='Training acc')
    axL.plot(val_acc,label='Validation acc')
    axL.set_title('Accuracy')
    axL.legend(loc='best')

    axR.plot(loss,label='Training loss')
    axR.plot(val_loss,label='Validation loss')
    axR.set_title('Loss')
    axR.legend(loc='best')

    plt.show()

#モデルを訓練する
def model_train(X,y,X_t,y_t):
    model = build_model(X.shape[1:])
    history = model.fit(X,y,batch_size=8,epochs=10,validation_data=(X_t,y_t))
    draw_graph(history)
    #モデルを保存する
    hdf5_file = 'model/spe^model.hdf5'
    model.save_weights(hdf5_file)
    return model

#モデルの評価
def model_eval(model):
    X = np.load('data/X_test.npy')
    y_test = np.load('data/y_test.npy')
    #データの正規化
    X_test = X.astype('float') /256
    y_test = np_utils.to_categorical(y_test,nb_classes)
    score = model.evaluate(x=X_test,y=y_test)
    print('loss : ',score[0])
    print('accuracy : ',score[1])

#学習開始
def main():
    X_train,X_test,y_train,y_test = np.load('data/train.npy')
    print('X_train shape : ',X_train.shape)
    print('X_test shape : ',X_test.shape)
    print('y_train shape : ',y_train.shape)
    print('y_test shape : ',y_test.shape)
    #データの正規化
    X_train = X_train.astype('float') /256
    X_test  = X_test.astype('float') /256
    y_train = np_utils.to_categorical(y_train,nb_classes)
    y_test = np_utils.to_categorical(y_test,nb_classes)
    #モデルを訓練し評価する
    model = model_train(X_train,y_train,X_test,y_test)
    model_eval(model)
if __name__ == "__main__":
    main()
```

kmeans.py

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

# csv 読込
df=pd.read_csv('abc_dataset.csv',encoding='Shift-JIS')
df.head()

#「標準化
scaler = StandardScaler()
df_sc = scaler.fit_transform(df.loc[:, ['A', 'B', 'C', '1to2', '2to3']])

X = pd.DataFrame(df_sc,columns=['A', 'B', 'C', '1to2', '2to3'])

# モデルの定義
clusters=6
kmeans = KMeans(n_clusters=clusters, random_state=0)
kmeans.fit(X)

labels = kmeans.labels_
result_df = pd.concat([X, pd.DataFrame(labels, columns=['kmeans_result'])], axis=1)

#csv
#result_df.to_csv("result_df.csv", index=False)

sns.pairplot(result_df, hue='kmeans_result', vars=['A', 'B', 'C', '1to2', '2to3']).savefig('pairplot_kmeans_result.png')
```

PCA.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

#csv 読込
df=pd.read_csv('abc_dataset.csv',encoding='Shift-JIS')
df.head()

#「標準化」or「正規化」
scaler = StandardScaler()
df_sc = scaler.fit_transform(df.loc[:, ['A', 'B', 'C', '1to2', '2to3']])

from pandas import plotting
plotting.scatter_matrix(df_sc.iloc[:, 1:], figsize=(8, 8), c=list(df_sc.iloc[:, 0]), alpha=0.5)
plt.show()

# 合成変数を 2 次元に指定
pca_model = PCA(n_components=2)

# データを学習して合成変数への変換行列を作成
pca_model.fit(df_sc)
transformed_variables = pca_model.transform(df.loc[:,  ['A', 'B', 'C', '1to2', '2to3']])

# データフレーム化
transformed_df = pd.DataFrame(transformed_variables, columns = ['pca_1', 'pca_2'])

transformed_df['label'] = df.loc[:,['label']]

transformed_df.head()

"""
#グラフ化
fig = plt.figure(figsize=(6, 6))

pca1 = transformed_df['pca_1']
pca2 = transformed_df['pca_2']
plt.scatter(pca1, pca2)
plt.legend()
plt.show()
"""

#正解ラベルに色付けグラフ化
fig = plt.figure(figsize=(6, 6))
for i in np.unique(df.loc[:,['label']]):
    pca1 = transformed_df['pca_1'][transformed_df['label'] == i]
    pca2 = transformed_df['pca_2'][transformed_df['label'] == i]
    plt.scatter(pca1, pca2, label=str(i))
plt.legend()
plt.show()

# 寄与率の表示
print(pca_model.explained_variance_ratio_)

# 合成変数間の相関係数の表示
print(transformed_df.corr())
```

PCAkmeans.py

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

#csv 読込
df=pd.read_csv('abc_dataset.csv',encoding='Shift-JIS')
df.head()

#「標準化」
scaler = StandardScaler()
df_sc = scaler.fit_transform(df.loc[:, ['A', 'B', 'C', '1to2', '2to3']])

#PCA
pca_model = PCA(n_components=2)

# データを学習して合成変数への変換行列を作成
pca_model.fit(df_sc)
transformed_variables = pca_model.transform(df.loc[:,  ['A', 'B', 'C', '1to2', '2to3']])

# データフレーム化
transformed_df = pd.DataFrame(transformed_variables, columns = ['pca_1', 'pca_2'])
tp_df1 = transformed_df[['pca_1']]
tp_df2 = transformed_df[['pca_2']]

#k-means
# モデルの定義
clusters=6
kmeans = KMeans(n_clusters=clusters, random_state=0)
kmeans.fit(tp_df2)

labels = kmeans.labels_
result_df = pd.concat([transformed_df, pd.DataFrame(labels, columns=['kmeans_result'])], axis=1)

#グラフ化
fig = plt.figure(figsize=(6, 6))
for i in range(clusters):
   pca1 = transformed_df['pca_1'][result_df['kmeans_result'] == i]
   pca2 = transformed_df['pca_2'][result_df['kmeans_result'] == i]
   plt.scatter(pca1, pca2, label=str(i))
plt.legend()
plt.show(
```