この課題における注意

発展課題は余裕があれば行ってください.

1 演習の目的 — 音源分離

この課題の目的は以下のとおりです.

i. 音声信号処理における基本的なテクニックについて学ぶ.
ii. 音源音声のもつ性質について学ぶ.
iii. 音源分離技術の原理を学び，実装する.

2 音源分離技術とは

近年，人が話した音声を機械が理解し，動作するシステムが増えてきています．例えば図 1.1 にあるように，スマートフォンに搭載されている音声アシスタントや，駅構内にあるスマートディジタルサイネージ，家庭で使用されるスマートスピーカなどがそれにあたります.

このようなシステムが普及した背景には，音声認識技術の躍進的な発展があります．深層学習を代表とする人工知能技術の成熟もあり，大規模なデータベースと高性能な計算機があれば，非常に高度な音声認識が実現できます．一般に音声ユーザーインターフェイス (Voice User Interface: VUI) を搭載した電子機器は，音声認識処理を大規模なクラウドサーバに任せて，エッジデバイス側では取り付けられたマイクロホンから音声を取得したり，サーバと通信を行います (図 1.2)．このような VUI 搭載デバイスは，サーバの大規模化やデバイスの省電力・省スペース化や，最近の次世代通信規格 5G による高速大容量通信の実用化などの要因によって，さらなる普及が見込まれています.

VUI はインターフェイスとして利便性が高い一方，聞きたい人の音声 (目的音声) 以外の音がマイクロホンに入ってくると，どうしても機械が目的音声の内容を理解することは困難になります．例えば駅構内で多くの人が話している中でスマートディジタルサイネージを利用する場合や，会議で多数の人が話している音声を音声認識で議事録を作成する場合などでは，目的音声のみを認識することは難しくなります (図 1.3).

このような場合，マイクロホンに入ってくる様々な音声が混じった音 (混合音) から目的音声のみを取り出

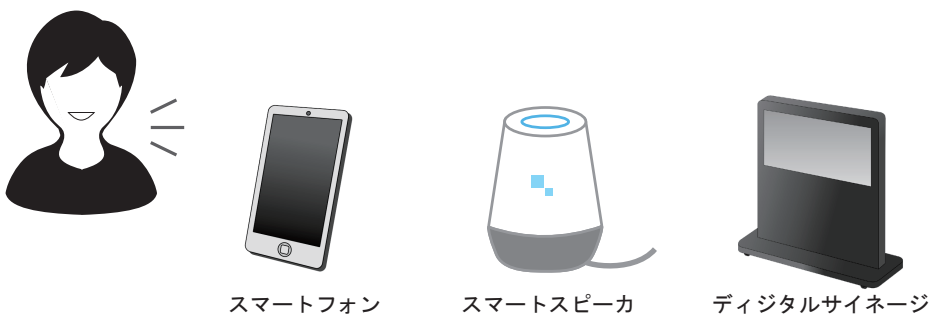

図 2.1 音声ユーザーインターフェイス (VUI) を搭載した機器

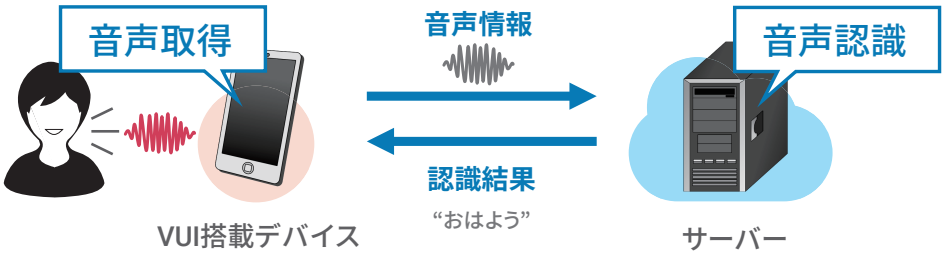

図 2.2 VUI 搭載デバイスを用いた音声認識処理の様子

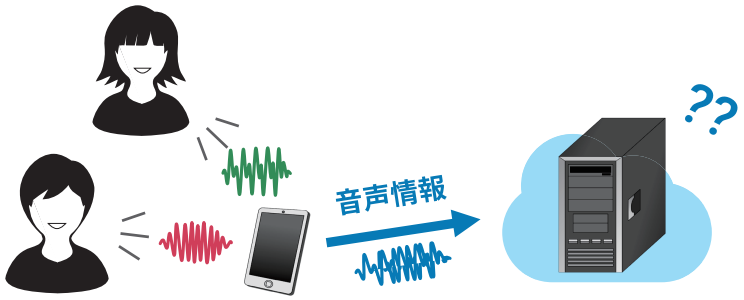

図 2.3 多数の話者がいる状況での音声認識

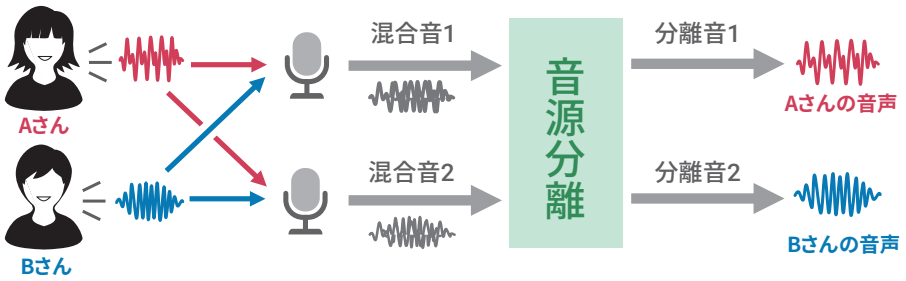

図 2.4 音源分離の概要図

す技術，すなわち**音源分離技術**が利用されます．最近のデバイスで利用される音源分離技術は，複数マイクロホン (マイクロホンアレイ) を用いる方法が主流となっています．図 1.4 に，マイクロホンが 2 つのときの音源分離システムの例を示します．同図のように，マイクロホンに話しかけている音声 (音源) は 2 つであると仮定します．マイクで取得された混合音は 2 つあります．音源分離システムではそれぞれの混合音を入力として，2 つの分離音声を出力とします．音源分離では，分離音声のうち，どちらかの音声が目的音声と一致するようにシステムのアルリゴリズムを設計することが課題となります．

この章のポイント

1. 音源分離は音声認識等のボイスユーザーインターフェイスのフロントエンドで使用される．

3 音声信号処理の基本

音源分離技術を学ぶ前に，音声信号処理の基本について学びましょう．音声信号処理では信号を**フレーム**と呼ばれる短い区間ごとに分割し，処理を行います．課題 A-1 で学んだ**スペクトログラム**も，音声をフレームに分割して振幅スペクトルを計算したもので，これはフレーム分割処理という基本手順に即したものになっています．音声信号処理においては，フレームの長さ (サンプル数) は20〜50msとなるように設定することが多いです．図 3.1 に音声信号処理の流れを載せます．

3.1 フィルタリング

分割した信号から欲しい信号を取り出す処理を**フィルタリング**と呼びます．フィルタリングは信号と**フィルタ**の内積によって実現されます．

3.1.1 周波数領域フィルタリング

以前に学んだように，フィルタリングは周波数領域で計算することもできます．この周波数領域フィルタリングは計算が簡便なため，音声信号処理では一般的な処理となっています．

周波数領域フィルタリングではまず分割信号を離散フーリエ変換します．ここでフレーム番号 i における分

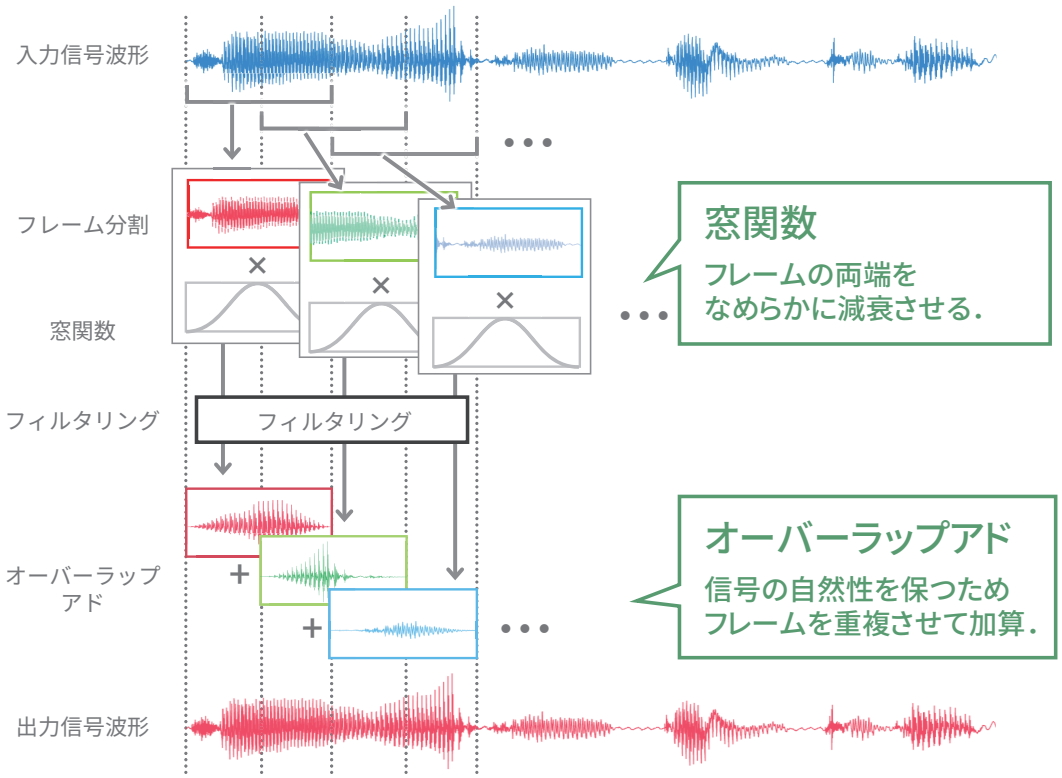

図 3.1 音声信号処理の概要図

割信号を $x_{i,n}$ (n はサンプル番号)，フィルタ係数を $h_{i,n}$，分割信号をフィルタリングした信号 $y_{i,n}$ とします．さらにそれらを離散フーリエ変換したものをそれぞれ $X(i,\omega)$，$H(i,\omega)$，$Y(i,\omega)$ とします．このとき，フィルタリングした分割信号 $y_{i,n}$ の離散フーリエ変換 $Y(i,\omega)$ は次の式のように計算されます．

$$Y(i,\omega) = X(i,\omega) \cdot H(i,\omega) \tag{3.1}$$

すなわち，$Y(i,\omega)$ は単に $X(i,\omega)$ と $H(i,\omega)$ の乗算によって計算することができます．フィルタ $H(i,\omega)$ が 0，または 1 の値をもつとき，$H(i,\omega)$ は**周波数マスク**とも呼ばれます．最後に $Y(i,\omega)$ を逆離散フーリエ変換して目的の分割信号波形を得ることができます．

3.2 オーバーラップアドと窓関数

フレームごとに分割して処理された信号は，再び時系列につなぎ合わせることで音声信号として復元することが出来ます．ただし，処理した信号をそのまま並べただけでは，フレーム間のつなぎ目で音声信号が不連続になる可能性があります．もし信号が不連続になると，復元音声は耳障りで不快なものになります．そこで信号の不連続性を解消するために，「**窓関数**」と「**オーバーラップアド (Overlap-Add)**」というテクニックを使用します．

3.2.1 窓関数

窓関数は山なりな形状をもつ関数で，フレーム分割された信号に掛け合わせることで両端をなめらかに減衰させることができます．話が前後しますが，窓関数はフレーム分割した後，フィルタリングを行う前にかけ合わせます．窓関数には形状が異なる数種類のものが存在し，音声信号処理においては一般にハミング窓 (Hamming Window) やハニング窓 (Hanning Winsow) が用いられます．

ハミング窓：

$$w(n) = 0.54 + 0.46\cos\left(\frac{n}{N}\pi\right) \quad (-N \le n \le N) \tag{3.2}$$

ハニング窓：

$$w(n) = 0.5 + 0.5\cos\left(\frac{n}{N}\pi\right) \quad (-N \le n \le N) \tag{3.3}$$

3.2.2 オーバーラップアド

窓関数をかけた信号を単に端同士でつなぎ合わせていくと，結合する境界の周辺で信号の振幅が極端に小さくなり，音声の自然性が失われます．そこで，フレームの一部を重ね合わせるように足していく，**オーバーラップアド**により音声の振幅をなめらかにするテクニックがあります．オーバーラップアドを使用する際には，フレーム分割の際にあらかじめ各フレームが隣接するフレームと重なるよう，冗長なフレーム分割を行います．

オーバーラップさせる割合はフレーム長の半分 (=ハーフオーバーラップ) が一般ですが，3/4 オーバーラップや 7/8 オーバーラップなど，オーバーラップの割合を増やしたやり方も存在します．**フレーム長とオーバーラップ幅は処理した音声の品質に影響を与える重要なパラメータです**．音声信号処理では一般に 7/8 オーバーラップが最も音質がよいとされています．ここで，3/4 オーバーラップではフレームのシフト幅 (=隣接フレームとの距離) はフレーム長の 1/4 になり，7/8 オーバーラップではフレームのシフト幅はフレーム長の 1/8 になります．

3.3 補足：Python における音声信号処理

Python では，音声信号処理を簡潔に実装するためのモジュールが多数公開されています．例えばフレーム分割，窓関数，および分割信号に対する離散フーリエ変換 (=短時間離散フーリエ変換，Short-Time Fourier Transdorm : STFT) の処理は，scipy モジュールの関数 `scipy.signal.stft` により一括して行うことができます．また，逆 STFT(=Inverse STFT ： ISTFT) およびオーバーラップアドは関数 `scipy.signal.istft` により一括して行うことができます．

したがってこれらの関数を用いれば，実は上に書いたようなややこしい (ように見える)フレーム分割や STFT，オーバーラップアドなどを考えなくとも音声信号処理を実現することができます．ただし，関数のパラメータを正しく与えなければ，意図した処理を実行することはできません．各々の関数の使い方については公式ドキュメントページをよく見て理解しましょう．

この章のポイント

1. 音声信号処理は一般にフレーム単位で行われる．
2. 分割した各信号に対して周波数領域フィルタリングを行い，欲しい信号を取り出す．
3. 処理したフレーム単位の音声信号をつなぎ合わせる際に，音声の自然性を保つために「窓関数」，「オーバーラップアド」が利用される．

演習 3-1：音声信号処理の基本　(低域通過フィルタリング)

1. ソースコード kadai_2-1.py をもとに 1000Hz 以下を通過させる低域通過フィルタを実装し，音声信号に対してフィルタリングを行え．ただし，フレーム長を 1024 サンプル，オーバーラップ長を 512 サンプルに設定せよ．
   - ソースコード kadai2-1.py をよく読み，実行し，音声処理の流れについて理解せよ．
   - 周波数マスクを修正し，所望の低域通過フィルタを実現せよ．
2. カットオフ周波数を 1000Hz から 500Hz に変更し，スペクトログラム，波形の変化を確認せよ．また，音声を聞いて聴覚上の変化を確認せよ．カットオフ周波数とは，フィルタリングにより抽出・除去する周波数の境界のことを指す．

補足：

低域通過フィルタのカットオフ周波数を低く設定すると，処理音声はこもったような音質になります．

3.4 バイナリマスクを用いた音源分離

次に，音源分離の中で最も単純な手法である「バイナリマスクを用いた音源分離」の原理を学習しましょう．今，図 3.2 のように話者が 2 名いてマイクロホンが 2 つある状況を考えます．A さんに着目すると，A さんの音声はマイクロホン 1 , マイクロホン 2 のどちらにも入力されます．ただし A さんとマイクロホンとの距離は異なります．ですので，近くにあるマイクロホン 1 に入力される音声のパワーは大きくなり，遠くにあるマイクロホン 2 に入力される音声のパワーは小さくなります (図 1.7)．一方で B さんの音声はその逆に，遠くにあるマイクロホン 1 に入力される音声のパワーは小さくなり，遠くにあるマイクロホン 2 に入力される音声のパワーは大きくなります．

この性質を利用して音源分離を行います．マイクロホン $i(i = 1, 2)$ に入力されるある時刻 t のある周波数 ω における振幅スペクトルを $X^{(i)}(t, \omega)$ と表現します．WDO を仮定すると，$X^{(i)}(t, \omega)$ は A さんあるいは B さんの音声の成分ということになります．$X^{(i)}(t, \omega)$ がどちらの音声に含まれるかを判定するために，$X^{(1)}(t, \omega)$ と $X^{(2)}(t, \omega)$ の大きさを比較します．すなわち，以下の通りに音声を振り分けます．

$$|X^{(1)}(t, \omega)| > |X^{(2)}(t, \omega)| \Rightarrow X^{(1)}(t, \omega), X^{(2)}(t, \omega) \text{ は } A \text{ さんの音声に含まれる．}$$
$$|X^{(1)}(t, \omega)| \leq |X^{(2)}(t, \omega)| \Rightarrow X^{(1)}(t, \omega), X^{(2)}(t, \omega) \text{ は } B \text{ さんの音声に含まれる．}$$

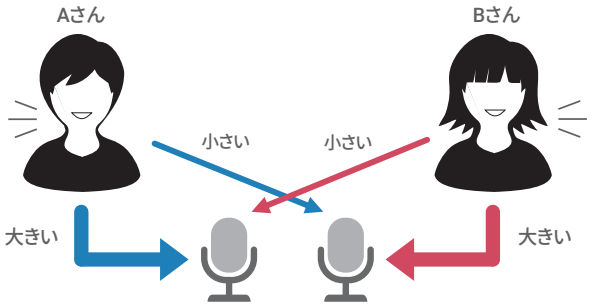

図 3.2 2 話者，2 マイクロホン環境の音源分離

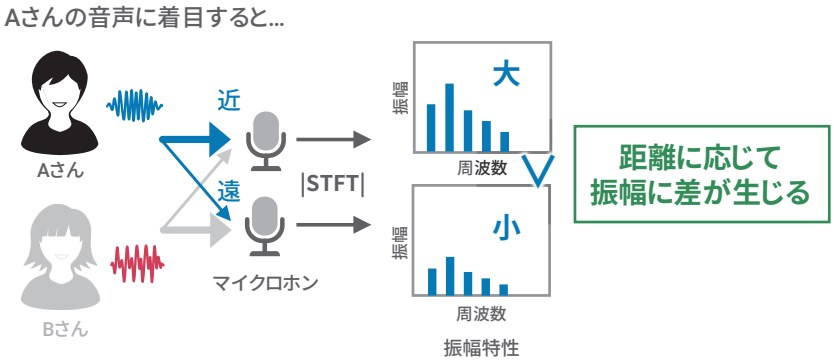

図 3.3 マイクロホンと話者間距離に応じた振幅の変化について

この処理を数式を用いて表現すると，以下の通りになります．

バイナリマスキング

$$\text{マスク}1：\ M_A(t,\omega) = \begin{cases} 1 & ,|X^{(1)}(t,\omega)| > |X^{(2)}(t,\omega)| \\ 0 & ,|X^{(1)}(t,\omega)| \leq |X^{(2)}(t,\omega)| \end{cases}, \tag{3.4}$$

$$\text{マスク}2：\ M_B(t,\omega) = \begin{cases} 0 & ,|X^{(1)}(t,\omega)| > |X^{(2)}(t,\omega)| \\ 1 & ,|X^{(1)}(t,\omega)| \leq |X^{(2)}(t,\omega)| \end{cases}, \tag{3.5}$$

$$A\text{ さんの周波数スペクトル：}\ X_A(t,\omega) = M_A(t,\omega) * X^{(1)}(t,\omega), \tag{3.6}$$

$$B\text{ さんの周波数スペクトル：}\ X_B(t,\omega) = M_B(t,\omega) * X^{(2)}(t,\omega) \tag{3.7}$$

ここで「*」は要素積を意味します．M_A, M_B は各時刻，各周波数で 0 または 1 の値をもつ行列で，入力信号のスペクトログラムとの要素積を計算することで，1 の値をもつ周波数成分のみを取り出す働きを持ちます．2 値の値 (バイナリ) で周波数の取捨選択を行うことから，これらの行列は**バイナリマスク**と呼ばれます．M_A は A さんの音声を，M_B は B さんの音声を取り出すためのバイナリマスクになります．

3.5 補足：音源分離性能の客観評価

音源分離におけるパラメータ (例えばフレーム長やオーバーラップ長など) を調整する場合，どれだけ音源分離が成功しているか，を客観的に評価したい場合があります．もし音源分離後の理想的な音声が入手できる場合，音源分離性能を客観的に評価する指標として，音声品質の近く評価 (Perceptual Evaluation of Speech Quality : **PESQ**) が用いられます．

PESQ は，「理想とする音声」と「評価したい音声」の 2 つを比べ，聴覚的にどれだけ近いかを数値化したものです．PESQ は-0.5 から 4.5 までの値を取り，大きいほど 2 つの音声が近いことを示します．PESQ は人間の主観的評価に非常に近い評価方法として知られています．

この章のポイント

1. ある話者が発した音声が複数のマイクロホンに入力されとき，各々の音声パワーはマイクロホンの距離に応じて変化する．
2. 各マイクロホンに入力される音声の振幅スペクトルの大小関係を調べることで，音源分離を達成するバイナリマスクを生成することが出来る．

演習 3-2 : バイナリマスクを用いた音源分離

1. 2 話者，2 マイクロホン環境におけるバイナリマスクを用いた音源分離を実装せよ．ただし，フレーム長を 1024 サンプル，オーバーラップ長を 512 サンプルに設定せよ．
   - ソースコード kadai_2-2.py を実行し，ステレオ音声に対する処理の流れを確認せよ．
   - バイナリマスクを実装し，音源分離を達成せよ．

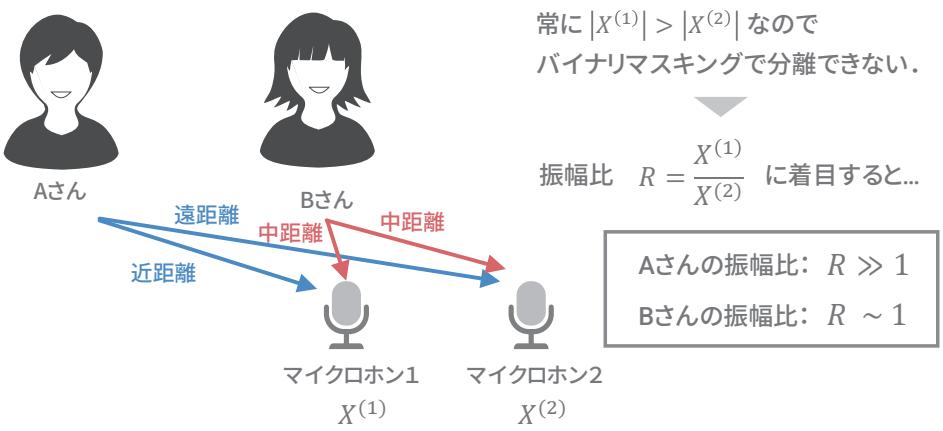

図 3.4 バイナリマスクを用いた音源分離が有効でない状況

3.6 DUET を用いた音源分離

バイナリマスクを用いた音源分離では，マイクロホン正面方向に対して各話者が左右にいる状況を想定していました．例えば図 3.4 のように，２名の話者がマイクロホン正面方向に対して右側，あるいは左側に偏っている場合，左右のマイクロホンで振幅に大小の違いが生まれず，バイナリマスクでは分離することはできません．このような状況での音源分離には，Degenerate Unmixing Estimation Technique, 通称 **DUET** が有効です．

3.6.1 DUET

DUET では，バイナリマスキングと同様，各マイクロホンに入力される音声の振幅比に着目します．振幅比は，マイクロホン 1 に入力する音声の振幅スペクトル $|X^{(1)}(t,\omega)|$ とマイクロホン 2 に入力する音声の振幅スペクトル $|X^{(2)}(t,\omega)|$ の比によって計算されます．数式で表すと，振幅比は以下の通りになります．

$$R(t,\omega) = |X^{(1)}(t,\omega)|/|X^{(2)}(t,\omega)| \tag{3.8}$$

図 3.5 は図 3.4 における音声を例として，各話者における振幅比のヒストグラムを求めたものです．横軸が振幅比で，縦軸が頻度です．同図からわかるように，話者 A(=音源 A) においては振幅比が最頻値の 2.0 に集中しています．一方で話者 B(=音源 B) においては振幅比が最頻値の 1.3 に集中しています．このように，振幅比は話者の位置によって異なります．またこの値は，話者の場所が変わらない限り，時間に依らず一定になります．

DUET では，この振幅比 $R(t,\omega)$ の特性を利用します．仮に音源 A における振幅比のヒストグラムの最頻値を $\bar{R}_A$，音源 B における振幅比のヒストグラムの最頻値を $\bar{R}_B$ としましょう．音源 A は，その振幅比の大部分が $\bar{R}_A - \sigma \sim \bar{R}_A + \sigma$ の範囲に含まれると考えます．一方で音源 B は，振幅比の大部分が $\bar{R}_B - \sigma \sim \bar{R}_B + \sigma$ の範囲に含まれると考えます．ここで σ は小さな定数です．すると，DUET による音源分離は以下の式で実現します．

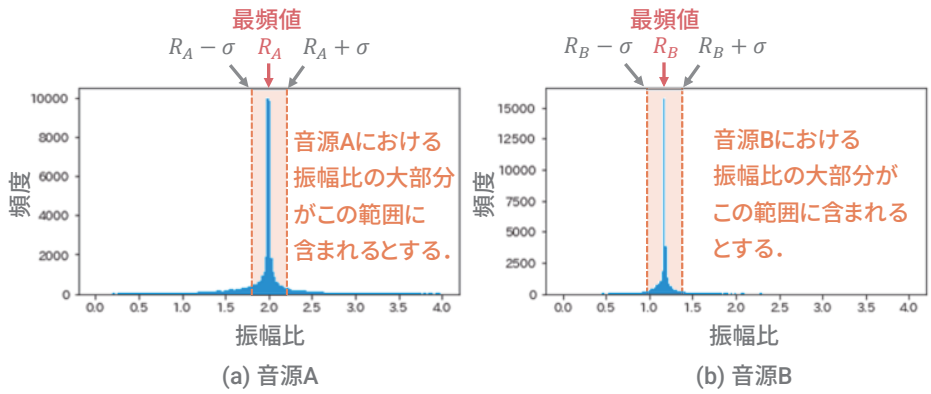

図 3.5 振幅比のヒストグラム

DUET

$$\text{マスク}1:\quad M_A(t,\omega)=\begin{cases}1 & ,\bar{R}_A-\sigma<R(t,\omega))<\bar{R}_A+\sigma\\ 0 & ,\text{その他}\end{cases}, \tag{3.9}$$

$$\text{マスク}2:\quad M_B(t,\omega)=\begin{cases}1 & ,\bar{R}_B-\sigma<R(t,\omega))<\bar{R}_B+\sigma\\ 0 & ,\text{その他}\end{cases}, \tag{3.10}$$

$$A\text{ さんの周波数スペクトル:}\quad X_A(t,\omega)=M_A(t,\omega)*X^{(1)}(t,\omega), \tag{3.11}$$

$$B\text{ さんの周波数スペクトル:}\quad X_B(t,\omega)=M_B(t,\omega)*X^{(2)}(t,\omega) \tag{3.12}$$

ただしこの処理を実現するためには，各音源における振幅比のヒストグラムの最頻値 $\bar{R}_A$，$\bar{R}_B$ を知っておく必要があります．すなわち，事前に A さんのみの音声，B さんのみの音声を取得した上でヒストグラムを計算し，$\bar{R}_A$，$\bar{R}_B$ を推定する必要があります．

3.6.2 改良 DUET

実際の音声で式 (3.9)-(3.12) の音源分離処理を行うと，耳障りが悪く，案外と音質が低くなります．なぜならば，実際の音声では話者の周波数成分が重複し，振幅比 $R(t,\omega)$ が $\bar{R}_A$，$\bar{R}_B$ 以外の値になる場合があるためです．そこで，より柔軟な音源分離を実現できるよう，しきい値を用いてマスクを作成します (図 3.6).

改良 DUET

$$\text{マスク}1:\quad M_A(t,\omega)=\begin{cases}1 & ,R(t,\omega)>R_T\\ 0 & ,\text{その他}\end{cases}, \tag{3.13}$$

$$\text{マスク}2:\quad M_B(t,\omega)=\begin{cases}1 & ,R(t,\omega)\le R_T\\ 0 & ,\text{その他}\end{cases}, \tag{3.14}$$

$$A\text{ さんの周波数スペクトル:}\quad X_A(t,\omega)=M_A(t,\omega)*X^{(1)}(t,\omega), \tag{3.15}$$

$$B\text{ さんの周波数スペクトル:}\quad X_B(t,\omega)=M_B(t,\omega)*X^{(2)}(t,\omega) \tag{3.16}$$

ただし，しきい値 R_T は以下の式を満たすよう設定する必要があります．

$$\bar{R}_B<R_T<\bar{R}_A \tag{3.17}$$

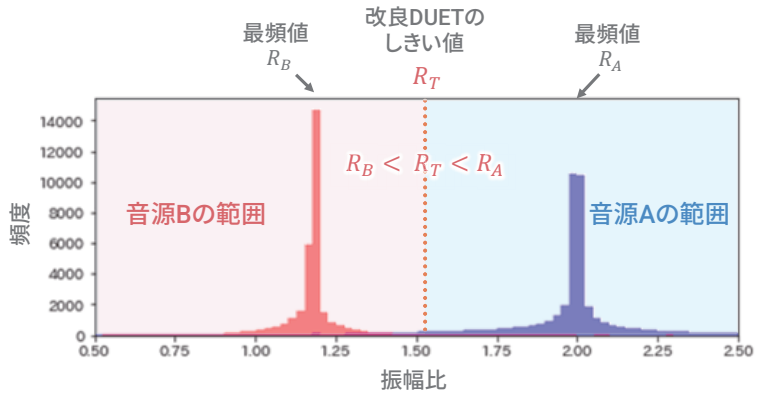

図 3.6 振幅比のヒストグラムと改良 DUET におけるしきい値の関係 (青：音源 A, 赤：音源 B)

この式では $\bar{R}_A$ が $\bar{R}_B$ より大きくなると仮定しています．この改良 DUET においても，$\bar{R}_A$，$\bar{R}_B$ はあらかじめ推定しておく必要があります．

3.6.3 改良 DUET とバイナリマスクの関係

実は，バイナリマスクは改良 DUET の 1 つのバリエーションに含まれます．式を見比べるとわかるように，バイナリマスクは $R_T = 1$ とした改良 DUET と完全に一致します．

この章のポイント

1. マイクロホン正面に対して話者が左側，あるいは右側に偏っている場合，DUET が有効である．
2. DUET では，マイクロホン間の振幅比を用いて音源分離用のマスクを生成する．
3. マイクロホン間の振幅比は，話者ごとに異なる値を示す．

演習 3-3 : DUET を用いた音源分離

1. 式 (3.9)-(3.12) を利用して，2 話者，2 マイクロホン環境における DUET を用いた音源分離を実装せよ．ただし，フレーム長を 1024 サンプル，オーバーラップ長を 512 サンプルに設定せよ．
   - ソースコード kadai_2-3.py を修正し，音源ごとに振幅比のヒストグラムを計算し，最頻値 $\bar{R}_A$，$\bar{R}_B$ を求めよ．
   - 求めた $\bar{R}_A$，$\bar{R}_B$ をもとにマスクを生成し，DUET による音源分離を達成せよ．ただし，$\sigma = 0.1$ と設定せよ．
2. 式 (3.13)-(3.16) を利用して，2 話者，2 マイクロホン環境における改良 DUET を用いた音源分離を実装せよ．
   - 推定した $\bar{R}_A$，$\bar{R}_B$ から任意に R_T を設定し，音源分離を達成せよ．
   - R_T の値を変えて音源分離性能 (PESQ の値) がどう変化するか確認せよ．
   - (1) の DUET と比べてどれだけ PESQ 値が変化したか確認せよ．

3.7 発展課題：DUET を用いた音源分離の応用

これ以降は発展課題になります．余力のある方のみチャレンジしてください．

DUET は，2 つのマイクロホンに対して 2 つ以上の音源を分離することが可能です．ここでは例として，3 話者・2 マイクロホン環境を想定しましょう (図 3.7)．この状況でも同様に，事前に音源 A，B，C ごとに振幅比のヒストグラムを求めます．各音源におけるヒストグラムの最頻値 $\bar{R}_A$，$\bar{R}_B$，$\bar{R}_C$ はそれぞれ異なる値を示し，話者の位置関係から $\bar{R}_C < \bar{R}_B < \bar{R}_A$ の関係が成り立ちます．このときの DUET におけるマスクは以下の式で計算できます．

3 音源に対する改良 DUET

$$\text{マスク 1：}\quad M_A(t,\omega) = \begin{cases} 1 & ,R(t,\omega) > R_{T_1} \\ 0 & ,\text{その他} \end{cases}, \tag{3.18}$$

$$\text{マスク 2：}\quad M_B(t,\omega) = \begin{cases} 1 & ,R_{T_2} < R(t,\omega) \leq R_{T_1} \\ 0 & ,\text{その他} \end{cases}, \tag{3.19}$$

$$\text{マスク 3：}\quad M_C(t,\omega) = \begin{cases} 1 & ,R(t,\omega) \leq R_{T_2} \\ 0 & ,\text{その他} \end{cases}, \tag{3.20}$$

$$A\text{ さんの周波数スペクトル：}\quad X_A(t,\omega) = M_A(t,\omega) * X^{(1)}(t,\omega), \tag{3.21}$$

$$B\text{ さんの周波数スペクトル：}\quad X_B(t,\omega) = M_B(t,\omega) * X^{(2)}(t,\omega), \tag{3.22}$$

$$C\text{ さんの周波数スペクトル：}\quad X_C(t,\omega) = M_C(t,\omega) * X^{(2)}(t,\omega) \tag{3.23}$$

ただし，しきい値 R_{T_1}，R_{T_2} は以下の式を満たすよう設定する必要があります．

$$\bar{R}_C < R_{T_2} < \bar{R}_B < R_{T_1} < \bar{R}_A \tag{3.24}$$

3.7.1 DUET の問題点

DUET は話者の数に制限がないので，非常に有用性の高い音源分離手です．一方で，話者の数が増加するにしたがって分離性能は低下します．実際に 3 話者の分離を行った音声を聞くと，音声の自然性が失われている

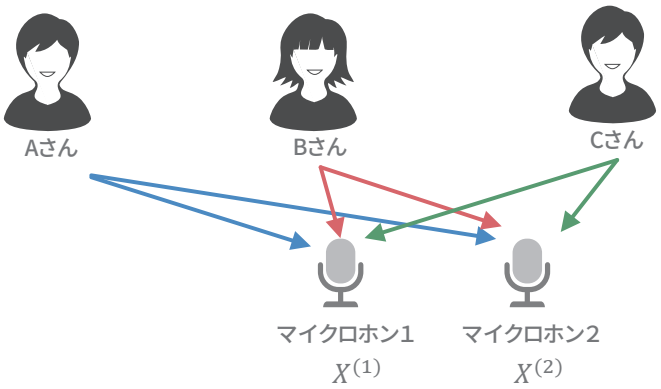

図 3.7 3 話者・2 マイクロホン環境

箇所があったり，分離に失敗している箇所があることがわかると思います．

これは，話者の数が増えるにしたがって，音声の周波数成分が重複しやすくなっていることが原因です．上記のような 0 または 1 の値をもつマスクでは，音声の周波数成分が重複していても，どれかの話者の音声として振り分けてしまっています．この問題を解決するためには，マスクに 0，1 以外の値も設定すればいいでしょう．例えば，「**振幅比に応じてマスクの値を変更する**」ことも有用でしょう．これを数式に落とし込んだ 1 例として，

$$\text{マスク}1: \quad M_A(t,\omega) = \begin{cases} 1 & ,R(t,\omega) > R_{T_1} \\ \frac{|X^{(1)}(t,\omega)|}{|X^{(1)}(t,\omega)|+|X^{(2)}(t,\omega)|} & ,\text{その他} \end{cases}, \tag{3.25}$$

$$\text{マスク}2: \quad M_B(t,\omega) = \begin{cases} 1 & ,R_{T_2} < R(t,\omega) \leq R_{T_1} \\ 0.1 & ,\text{の他} \end{cases}, \tag{3.26}$$

$$\text{マスク}3: \quad M_C(t,\omega) = \begin{cases} 1 & ,R(t,\omega) \leq R_{T_2} \\ \frac{|X^{(2)}(t,\omega)|}{|X^{(1)}(t,\omega)|+|X^{(2)}(t,\omega)|} & ,\text{その他} \end{cases}. \tag{3.27}$$

なども考えられます．

音源分離のアルゴリズムやパラメータの設定方法は様々考えられます．いろいろな方法を自身で編み出して，試してみましょう！

発展演習 3-4 : DUET を用いた音源分離の応用

この演習は必須ではありません．余力のある方のみチャレンジしてください！

1. 式 (3.18)-(3.23) を利用して，3 話者，2 マイクロホン環境における DUET を用いた音源分離を実装せよ．
2. 音声分離性能を向上させるための工夫を行え．
   - フレーム長，オーバーラップ長によって音源分離性能が変化する．フレーム長やオーバーラップ長を調整して，PESQ 値を向上させよう．
   - 振幅比に応じてマスクの値を変えるなど，様々な方法にチャレンジしてみよう．

参考文献

[1] 古井 貞熙 (著), “音声・音響工学”, 近代科学社, 1992/09 (ISBN : 7649-0196-X).

[2] 戸上 真人 (著), “Python で学ぶ音源分離（機械学習実践シリーズ）”, インプレス社, 2020/08 (ISBN : 978-4-295-00984-9).

[3] 青木 直史 (著), “ディジタル・サウンド処理入門”, CQ 出版社, 2006/04 (ISBN : 7898-3090-X).

[4] 川村 新 (著), 黒崎 正行 (著), “ディジタル音声＆画像の圧縮/伸張/加工技術”, CQ 出版社, 2013/05 (ISBN : 978-4-789-83145-1).