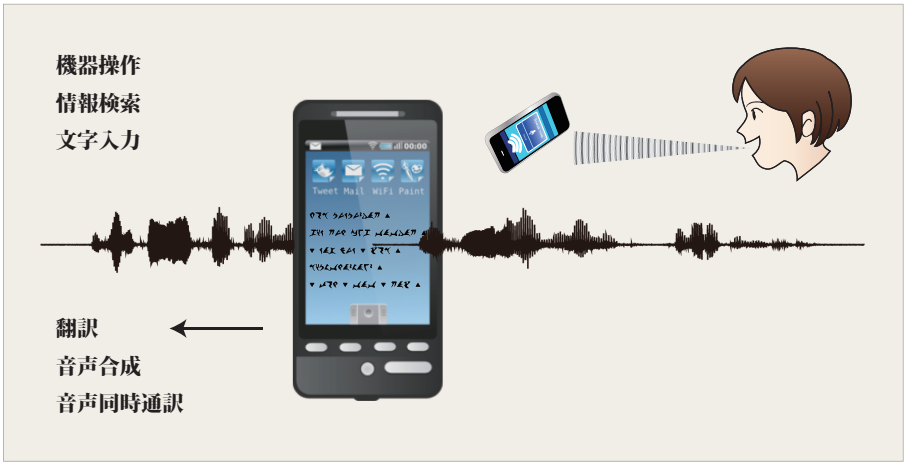
第1章　はじめに

1.1　音声認識とは

音声認識（speech recognition）とは，音声波に含まれる言語の意味内容に関する情報を機械（コンピュータ）で自動的に抽出することです．狭い意味では，音声の内容を自動的に文字に書き取ることをいいます．キーボードを打つ代わりに，コンピュータに話した内容がディスプレイに表示されるタイプライタのようなものといっていいでしょう．広い意味では，誰が話しているかなどの個人性情報といったものの抽出を行う話者認識などの技術を含みます．

音声認識には，あらかじめコンピュータに登録した単語やコマンドを認識する**単語認識（音声コマンド認識**とも呼ばれます）から，数千以上の単語を扱い単語ごとに区切らず普通に話された音声を認識する**大語彙連続音声認識（ディクテーション**とも呼ばれます）までさまざまなものがあります．



1.1.1　音声認識の難しさ [28]

音声認識を構成する中心的な技術要素は**パターン認識（pattern recognition）**です．音声認識はパターン認識の中でも最も難しいクラスの問題と位置付けられています．パターン認識の問題の難しさは，入力及び出力それぞれの複雑さと，その対応関係で定義されます．

パターン認識の問題における最も簡単なクラスは，「1 入力 1 出力パターン認識問題」で，枠内に書かれた文字を認識するような問題が典型的な例です．これに対して，音声は時系列信号なので，入力は複数（一定の時間間隔で区切られた特徴量の系列）であり，それらを単語の系列として認識することになるので出力も複数です．さらに，入力の系列長と出力の系列長の間に明確な対応関係がありま

せん．早口で話すのとゆっくり話すのとでは，単位時間の単語数が異なります．従って，音声認識はパターン認識の中で最も難しい問題である「系列長の対応関係が無い複数入力複数出力のパターン認識問題」なのです．

1.1.2 音声認識の歴史

音声認識の研究は 80 年以上前から行われていて，さまざまな技術が研究開発されてきました．文献 [23, 36] で解説されている研究開発の世代を紹介します．

1950 年から 1960 年代の第 1 世代においては，音声信号の物理的特徴に関する知見に基づいて，アナログフィルタバンクと論理回路を用いた経験則的な処理が用いられました．

1960 年から 1980 年代の第 2 世代では，音声認識に有効な音響特徴量と動的パターンマッチング手法に関する研究が進み，あらかじめ蓄えておいたテンプレートと DP マッチングに代表される時間軸整合マッチングが行われました．

1980 から 1990 年代の第 3 世代では，DP マッチングを拡張した形で隠れマルコフモデル（Hidden Markov Model: HMM）と，HMM の各状態の音響特徴量のパターンをモデル化する混合正規分布（Gaussian Mixture Model: GMM）が導入されました．このような音声モデルを GMM-HMM と呼びます．N-gram 言語モデルが導入され，その後の実用的音声認識システムの基盤技術となりました．

1990 年から 2000 年代は，誤り率最小化に基づく識別的モデル，モデル適応，話し言葉音声データベースなどによってこの基盤技術の高度化が進められました (第 3.5 世代)．

2010 年代以降は第 4 世代と呼ばれ, 深層ニューラルネットワーク（Deep Neural Network: DNN）によって大幅な性能向上が達成されています．音響モデルについては，GMM による確率計算を DNN に置き換えた DNN-HMM が，言語モデルについては，再帰型ニューラルネットワーク（Recurrent Neural Network: RNN）を N-gram と併用するモデルが一般的になっています [36]．近年では RNN を発展させた長・短期記憶（Long Short-Term Memory: LSTM）のみで直接音声認識結果を出力する End-to-End（端から端まで．つまり入力層に音声特徴量を入れると出力層に認識結果が出てくる．）が研究されています．最近では、大規模言語モデル（Large Language Model: LLM）の中核を成す Transformer により，自然言語だけではなく画像、音楽、音声などのマルチモーダル（multimodal、複数の種類のデータを統合して扱う）処理が可能になりました．この技術の応用として，例えば自動同時音声通訳の実用化が見通せる段階になっています．ChatGPT（Chat Generative Pre-trained Transformer）の最新版である ChatGPT-4o (omni）では人間同士の会話のような応答速度が高い自然な対話が可能となっています．

この実習では第 3 世代の GMM-HMM を用いた単語音声認識を行います．受講者が認識させたい単語の音声を自分の声で録音して学習用データを作り，GMM-HMM の音声モデルを学習させて，単語認識ができるような仕組みを作ります．この内容にした理由については第 7 章「おわりに」をご覧ください．

1.2　単語音声認識とは

この実験では，もっとも簡単で基本的な音声認識方式である単語音声認識[1]を課題として行います．単語音声を認識することができれば，たとえば，ウェブ資料の検索のためにキーワードを入力したり，カーナビに目的地の地名を入力することができます．

単語音声認識を実現するためには，まず，認識対象となる単語を認識するための音声パターンをコンピュータに登録します．例えば，カーナビに行き先を入力するための単語認識を行う場合は，「東京駅」「調布」「電通大」などの音声パターン $\boldsymbol{y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_K\}$ を登録しておきます．そして発話された単語をスペクトル分析して得られた音声パターン $\boldsymbol{x}$ と登録パターン $\boldsymbol{y}_k$ $(k = 1, 2, \ldots, K)$ との比較を行います．$d(\cdot, \cdot)$ を適当なスペクトル距離尺度とします．比較の結果，パターン間距離 $d(\boldsymbol{x}, \boldsymbol{y}_k)$ が最も小さいもの，すなわち $\underset{k}{\operatorname{argmin}}\, d(\boldsymbol{x}, \boldsymbol{y}_k)$ を認識結果とします[2]（図 1.1）．

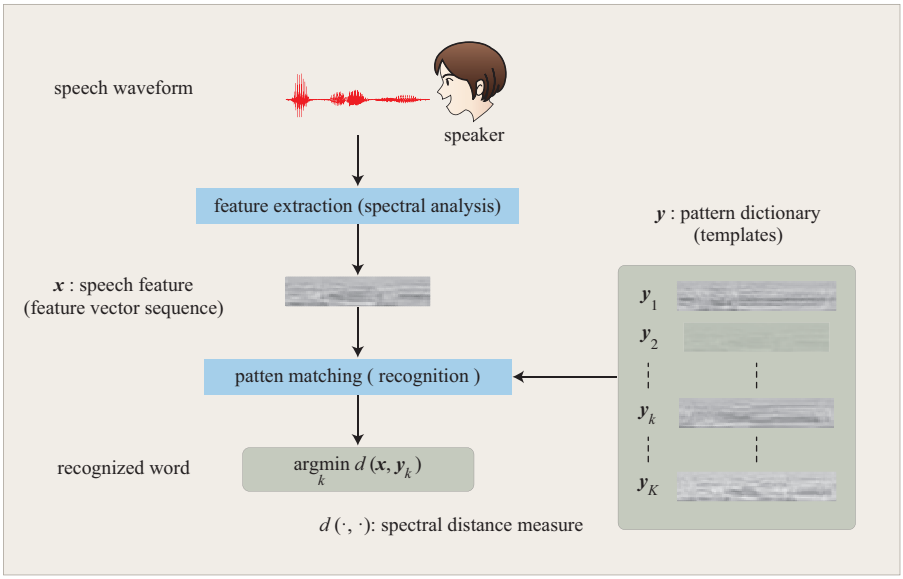


図 1.1: 音声のパターン認識（単語音声）．$\boldsymbol{x}$：音声スペクトル特徴パターン，$\boldsymbol{y}$：パターン辞書，$d(\cdot, \cdot)$：スペクトル距離尺度

発話された音声と登録パターンとの距離を計算する場合，単純に比較することはできるでしょうか？ 異なる話者が発話した「青い」/aoi/という単語の音声波形を比較してみます（図 1.2）．話者 K（図 1.2（a））に比べて，話者 F の発話（図 1.2（b））の継続長は 1 割ほど短いのですが，/o/ の継続長は逆に 5 ms 長くなっています（話者 K は 125 ms，話者 F は 130ms）．このため，話者 K の発話を単純に時間方向に 1 割縮めても，話者 F の発話内容と時間的に一致しません．この例のように，一般に 2 つの発話の長さは異なります．録音した音声を再生するのでない限り，発話の長さが同じになることはありません．さらに，単語に含まれる個々の音の長さは，音の種類によって伸縮のしかたが異なります．このた

[1] 音声認識の文脈では単語音声認識を単に単語認識と呼ぶこともあります．

[2] 距離ではなく類似度 $s(\cdot, \cdot)$ を用いる場合は，類似度 $s(\boldsymbol{x}, \boldsymbol{y}_k)$ が最も大きいもの，すなわち $\underset{k}{\operatorname{argmax}}\, s(\boldsymbol{x}, \boldsymbol{y}_k)$ を認識結果とします

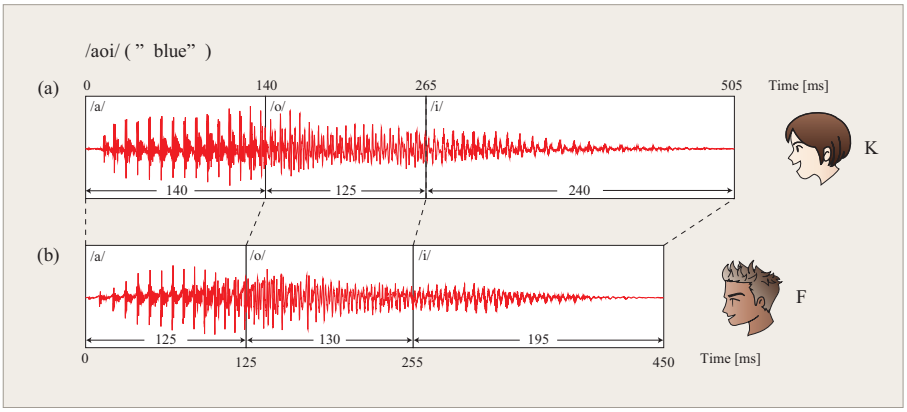


図 1.2: 音声パターンは非線型に伸縮する．「青い」/aoi/の話者 K と話者 F の音声波形，音素の位置と継続時間の比較.

め，音声パターンの比較は，時間軸方向に部分的に伸縮させながら行う必要があります．時間軸方向の非線型伸縮を行いながらパターン比較を行う古典的手法には，DP マッチングや隠れマルコフモデルがあります．本実験では隠れマルコフモデル（Hidden Markov Model: HMM）を用います.

1.3 概要

音声信号のスペクトル分析法，統計的音声認識の枠組みを理解し，音声認識とはどのような技術であるのかを知り，隠れマルコフモデル（HMM）の原理と算法を身につけ，自分の音声による単語認識を体験することによって，音声認識の実際について学ぶことが本実験テーマの狙いです.

第 2 章では音声の基本事項，すなわち声帯や舌などの音声を生成する器官，音声生成の仕組み，音声信号のスペクトルの性質，日本語の音素について概説しています．音声のスペクトル分析の結果をみるために必要な基礎知識です．第 6 章で単語音声データを作成する際の音声区間の始まりと終りを定めるときに役立つ知識です.

第 3 章では音声情報処理に必要な音声スペクトル分析について学びます．スペクトル分析では，音声に含まれているさまざまな高さの音の成分の強弱を抽出します．音声の振幅波形をフーリエ変換して得たパワースペクトルを人間の聴覚特性を模擬したフィルタバンクに入力し，その出力に対して離散コサイン変換を行い，メル周波数ケプストラム係数（MFCC）を得ます．この MFCC を音声の特徴量として用いて単語認識を行います．この章では，音声振幅波形から MFCC を得る計算手順を学び，C 言語でプログラミングを行い，音声の分析を行います.

第 4 章では，この実験の基礎となる統計的音声認識について説明します．これは現在一般に用いられている音声認識技術に共通する基礎原理です.

第 5 章では，HMM の原理を学びます．HMM をパターン認識器として用いる場合，入力パターンを生成する確率を効率的に計算する算法が必要です．そのための Forward アルゴリズム，Backward アルゴリズムの C 言語のプログラムを作り，コンピュータ上で実行してみます．また，HMM パラメータの学習に用い

られる Baum-Welch アルゴリズムを C 言語でプログラミングし，簡単なパターンの実例を用いて HMM を学習し，その HMM を使ってパターン認識を行ってみます．さらに，単語音声のモデル化に必要な多次元ガウス確率密度関数のプログラミングをします．

第 6 章では，単語認識を行うために必要な単語 HMM と，それを用いて単語認識を行う方法について学びます．受講者が自らの音声を使って単語音声認識を行います．自分の声を使って単語の音声データを収集し，そのスペクトルパターンから単語 HMM を学習します．学習した HMM の認識性能を測定した後，最後にマイク入力の音声を On-The-Fly 認識（入力された音声をその場で直ちに認識）する実験を行います．

1.4 必須条件

C 言語でプログラムを書き，コンパイルして実験用のプログラムを製作します．そのプログラムを用いて複数のデータに対する繰り返し計算を行うために，シェルプログラミングの機能を使います．そのため，この実験では，受講者が C 言語プログラミングの基本を理解し，実習した経験があること，Linux の CUI（文字ユーザインタフェース）すなわちキーボードからコマンドを入力して計算機を操作することに熟れていること，およびシェルの扱いを知っていることを前提としています．IED 教室の端末は Ubuntu で起動してください．実験を行うときは，まずデスクトップ上部右端のアイコンをクリックして「端末」（図 1.3）を起動してください．端末は複数起動しておいた方が作業が効率的です．ファイル操作，音声編集ソフトウェアやテキストエディタの起動，音声認識の実施は全てこの端末にコマンド入力することによって行います．

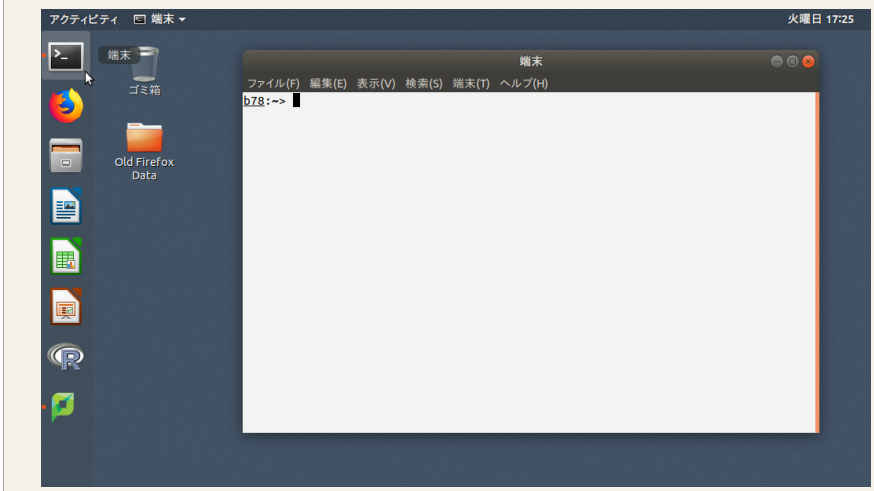


図 1.3: 実験の手順は端末へのコマンド入力によって進める．端末はデスクトップ上のアイコンをクリックして起動する．

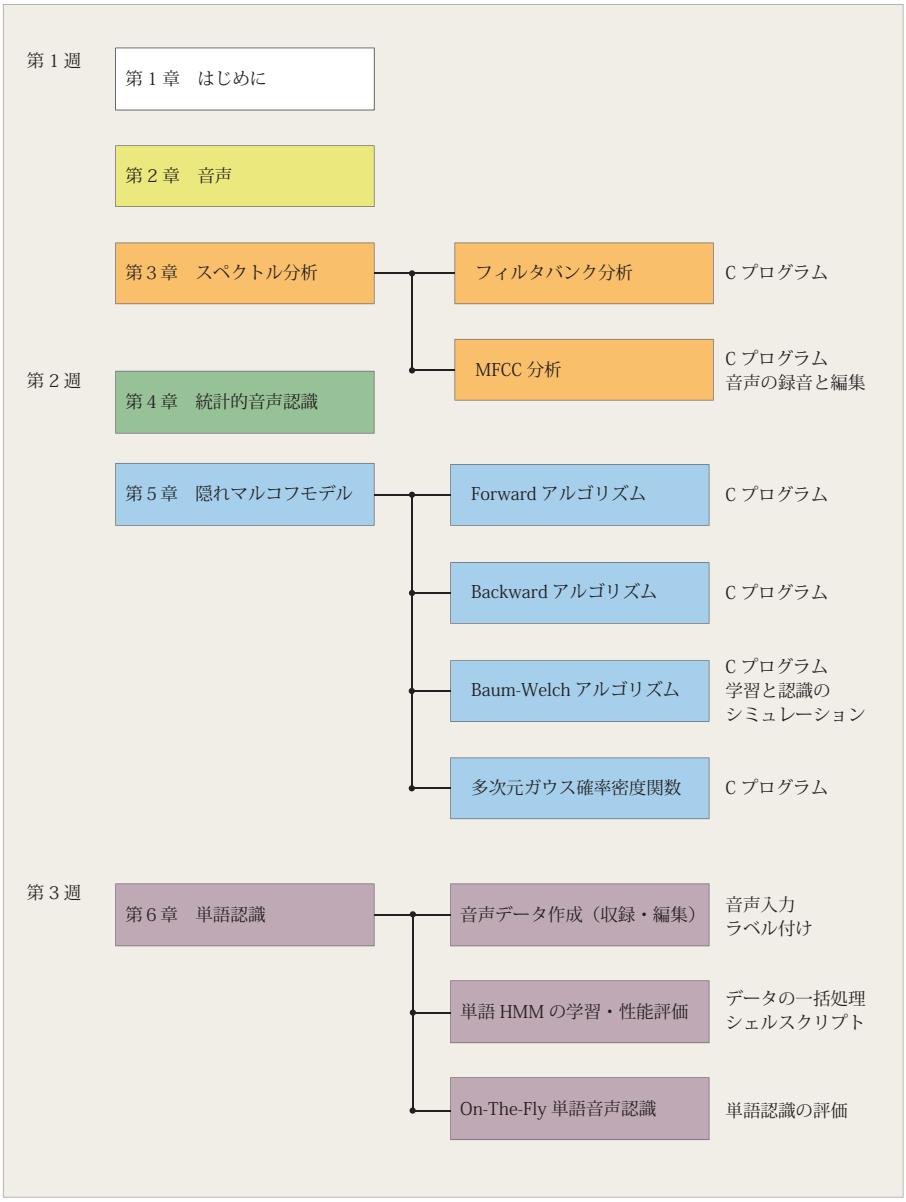


図 1.4: 実験書の構成と日程

1.5 日程

第 1 週では，音声信号のスペクトル分析について学び，C 言語でスペクトル分析のプログラムを作成します．そして，音声入力の練習を兼ねて自分の音声を録音し，スペクトル分析をしてみます（第 2 章，第 3 章）．

第 2 週では，HMM の基本算法である Forward アルゴリズムと Backward アルゴリズムのプログラムを作ります．また，音声のパターンをコンピュータに学習させるための HMM 算法である Baum-Welch アルゴリズムの実習，及び多次元ガウス確率密度関数のプログラミングを行います（第 4 章，第 5 章）．

第 3 週では，自分の音声を録音し，その音声を学習データとして用いて単語 HMM を学習し，それを用いて音声認識の実験を行います．まず，あらかじめ録音しておいた自分の音声を用いて認識率の評価をし，つぎに，マイクから入力した音声を直接認識することを試します（第 6 章）．

1.6 レポート

1.6.1 形式・期限

実験日の翌週の水曜日の 23:59:59 までに, asr@takagi.inf.uec.ac.jp に提出すること．毎回提出してください．レポートの形式は Portable Document Format（PDF）に限ります．PDF ファイルに対し，セキュイティ設定（ファイルロック，パスワード等）を行わないこと．PDF ファイル名は半角英数字で ”学籍番号_通し番号_姓”とすること（例：1234567_1_Takagi.pdf）．メールの Subject は PDF ファイルと同一とすること．

1.6.2 課題一覧

第 1 日 スペクトル分析：第 3.10 節

- フィルタバンク分析：第 3.10.1 節
- メル周波数ケプストラム分析：第 3.10.2 節
- 自分の声の録音，編集，分析：第 3.10.3 節

第 2 日 HMM 算法のプログラミングとシミュレーション：第 5.6 節

- 前向きパスアルゴリズム：第 5.6.1 節
- 後ろ向きパスアルゴリズム：第 5.6.2 節
- Baum-Welch アルゴリズム：第 5.6.3 節
- 多次元ガウス確率密度関数：第 5.6.4 節

第 3 日 単語音声認識：第 6.5 節

- 単語認識タスクの設計：第 6.5.1 節
- 単語 HMM の学習：第 6.5.4 節
- 学習の検証：第 6.5.5 節
- On-The-Fly 単語認識：第 6.5.6 節

1.6.3 穴埋め問題について

課題の中には C 言語のソースコードの一部が削除されていて，その部分を補って所望の処理を行うプログラムを完成させるものがあります．ソースコード中の穴埋め部分は`/* fill in blank */`で示しています．穴埋め部分は 1 行で示されていますが，補うべきコードは，代入文の右辺である場合や，ループや条件判断などの制御を伴う複数行のものであったりします．十分ご注意ください．

1.7 計算機実習の準備

1.7.1 ファイルのコピー

自分のホームディレクトリの直下に asr というサブディレクトリを作って，実験に必要なファイルを用意します．

```
[~]% cd
[~]% cp -pR ~takagi/asr ./
```

と入力すると，カレントディレクトリ直下に asr というサブディレクトリができます．念のため，上記を実行する前に同じ名前のディレクトリが無いことを確認してください．コピーされるファイルの内訳については、付録 B をご覧ください．

1.7.2 環境設定の追加

実習に必要なコマンドのパスを~/.cshrc に追加したり，音声編集プログラムの設定を変更するために，端末のコマンド入力で以下を実行してください．

```
[~]% tcsh -f ~/asr/addpath
[~]% source ~/.cshrc
```

この操作により，シェル環境設定ファイル$home/.cshrc が$home/.cshrc_backup_asr に退避保存され，この実習のためのパス設定などが$home/.cshrc に追加されます．元の環境に戻したい場合は，退避保存されたファイルに戻してください．

1.7.3 シェル環境についての注意

この実習の説明は tcsh 環境を前提としています．その他のシェルを使っている場合は，各自その環境に合わせた対応をしてください．

なお，このテキストのコマンド実行例は，カレントディレクトリが分かるように，シェルの設定ファイル~/.cshrc にコマンドプロンプトの形式を次のように設定した環境で作っています．

```
set prompt = "[%c2]% "
```

1.7.4 エディタについての注意

Ubuntu 付属の「テキストエディタ」は、本実習の単語認識実験に用いる設定ファイルの作成に適していないので，これ以外の開発環境やテキストエディタを使ってください．