第6章　単語認識

6.1　はじめに

単語を発声した音声を認識することを**孤立単語音声認識 (isolated word recognition)** といい，通常，これを単語音声認識（spoken word recognition）あるいは単に単語認識（word recognition）と呼びます．単語認識を応用して機械を声で操作することに使う文脈では音声コマンド認識（voice command recognition）と呼ぶこともあります．

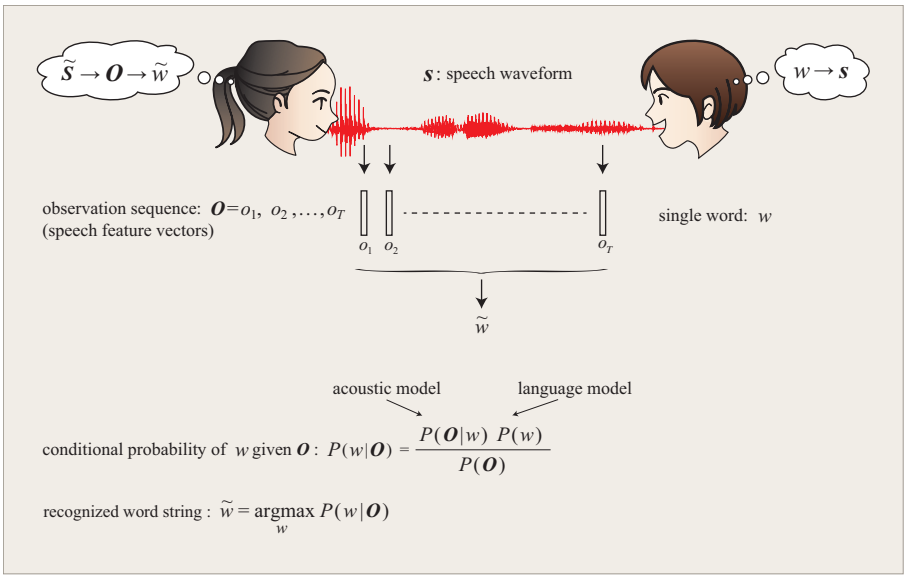


図 6.1: 単語認識の定式化．W：単語，S：音声波形，$\tilde{S}$：聴き手が受け取った音声波形，O：観測系列（聴き手の受け取った音声特徴の系列），$\tilde{W}$：推定単語

単語音声がスペクトル分析によって得た音声特徴ベクトル時系列（観測系列）を

$$\boldsymbol{O} = \boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T \tag{6.1}$$

とします．ここで，$\boldsymbol{o}_t\ (t = 1, \ldots, T)$ は時刻 t で観測された音声特徴ベクトルです．単語認識は，

$$\underset{i}{\operatorname{argmax}}\, P(w_i|\boldsymbol{O}), \quad w_i \in \mathcal{W} \tag{6.2}$$

を計算する問題と考えることができます．ここで，$\mathcal{W}$ は単語の集合（認識対象単語），$w_i\ (i = 1, \ldots, |\mathcal{W}|)$ は i 番目の単語を表わします．式 6.2 の確率は直接計算することはできませんが，Bayes の定理を用いて 4.2 節と同様に，

$$P(w_i|\boldsymbol{O}) = \frac{P(\boldsymbol{O}|w_i)P(w_i)}{P(\boldsymbol{O})} \tag{6.3}$$

と変形し．この式から計算することができます．

事前確率 $P(w_i)$ が与えられた条件のもとで，単語の発話された確率の計算は $P(\boldsymbol{O}|w_i)$ のみに依存します．観測系列 $\boldsymbol{O}$ の次元の大きさを考えると，条件付同時確率 $P(\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T|w_i)$ を単語音声のサンプルデータから直接計算するのは実際的に無理です．しかしながら，隠れマルコフモデル（Hidden Markov Model: HMM）などの単語音声生成モデルを用いれば，条件付き確率 $P(\boldsymbol{O}|w_i)$ の計算の問題を HMM の出力確率 $P(\boldsymbol{O}|\lambda)$ の計算の問題に置き換えることができます．

なお，この実習では，全ての単語は等確率で出現すると仮定します．すなわち，

$$P(w_i) \triangleq \frac{1}{|\mathcal{W}|}, \quad i = 1, \ldots, |\mathcal{W}|.$$

6.2 単語 HMM

隠れマルコフモデル（HMM）は，5 章の演習で取り扱ったような離散記号を出力することができるだけでなく，音声スペクトルのような連続値，連続値ベクトルを出力することもできます．HMM で単語音声をモデル化するためには，出力 $b_j(\boldsymbol{o}_t)$ として確率密度関数を定義します．連続 HMM の詳細は 5.4.2 節を参照してください．

単語音声は音素が決められた順に発声されたものです．たとえば，日本語の数詞の 1 は音素が/i/，/ch/，/i/の順に発声されて単語音声となります．順番を変えて/ch/，/i/，/i/と発声されると違う単語になってしまいます．このように，音の順番が単語の意味の違いに関わるので，単語をモデル化する HMM としては，状態遷移の方向に強い制約を掛けた一方向性 HMM（5.4.1 節）を用います．つまり，各状態の出力は音素（実際には音素よりも小さな音声単位）のスペクトルの確率分布であり，遷移先はその状態かあるいは後の状態に制限されています．前の状態に戻らないことにより，発音の時間進行の方向が決まるのです．

実習で用いる数詞の「1」の単語音声を例に説明します（図 6.2）．単語の前後に 100ms の無音区間を含んだ波形データを HMM で学習します．波形データをスペクトル分析した結果の MFCC には，3.8 節で述べたように局所的に特徴のあるパターンが見られ，音素（波形）と対応して変化していることがわかります．ちなみに，子音/ch/は 2 種類の音が連なっている音素です．前半に音声が一旦途切れる無音部（閉鎖部，`cl`）があり，後半に破裂部および摩擦部（`ch`）があります．母音（この単語では/i/）はその区間でほぼ一定の性質を示します[1]．

単語 HMM の状態数は，上記のような音素の性質や MFCC のパターンの変化のしかたを考慮して決めます．図 6.2 の例では，単語 1 の HMM の状態数を 7 としています．単語 HMM の学習では，まず，単語音声の MFCC の時系列 $\boldsymbol{O}$ を状態数で均等に分割し，時間順に状態に割り当てます．次に，状態に割り当てられた部分時系列について，各次元の係数の値の分布を混合正規分布で表現します（5.4.2 節）．この例では，MFCC の特徴のある各区間がほどよくそれぞれ 1 つの状態に割り当てられています．

[1] より正確には，前の音素からの過渡区間，定常部，後の音素への過渡区間という性質の異なる 3 つの部分からなりますが，/ch/のような子音と比べれば，定常であるといえます．この実習では個別の音素のモデル化は行わないので，このような詳細は気にしないことにします．

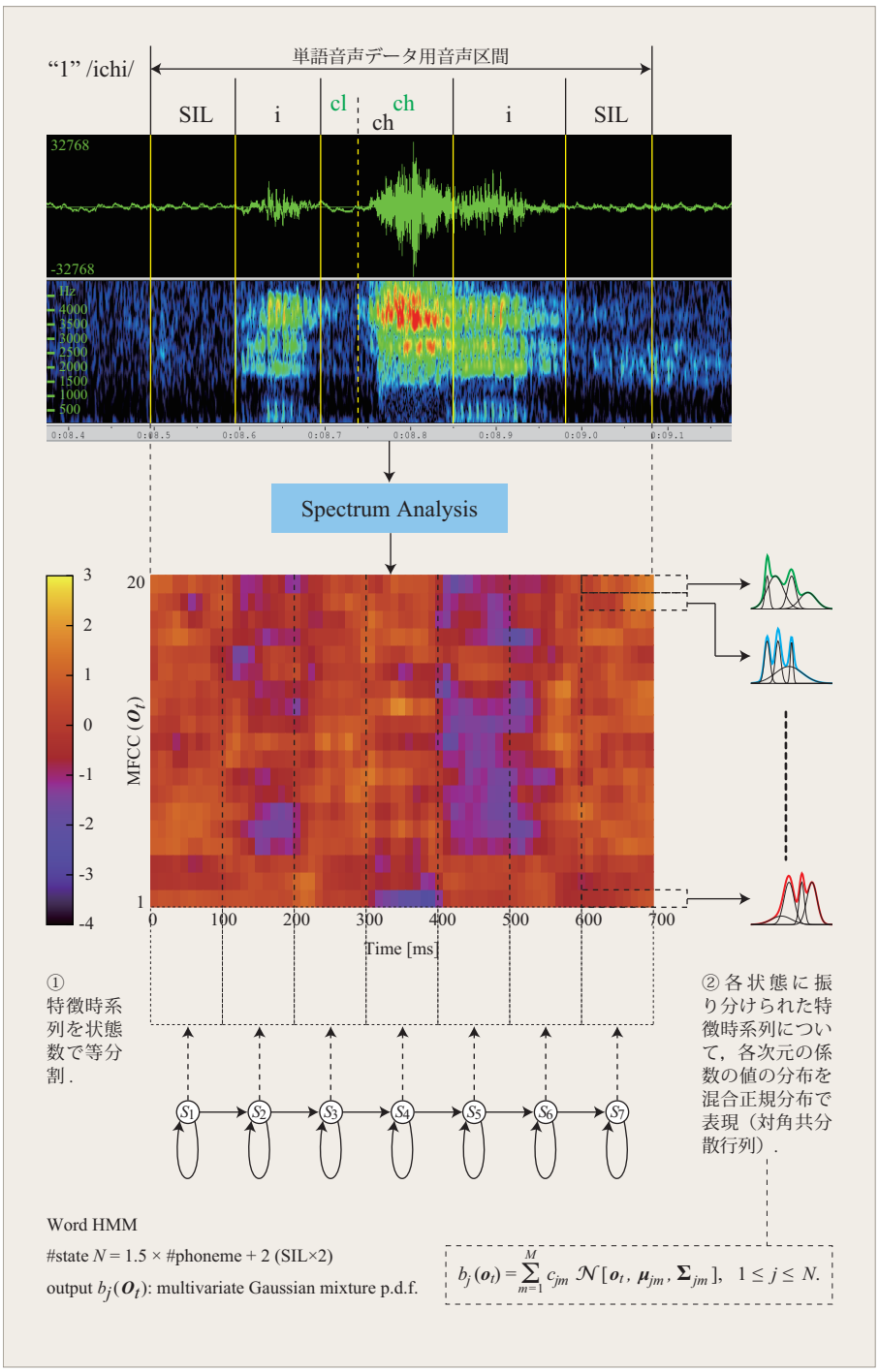


図 6.2: 単語音声スペクトルを生成する HMM．HMM の状態 S_i からの出力は，単語を構成する音の部品に対応します．出力 $b_j(\boldsymbol{o}_t)$ は多次元実数ベクトルである MFCC の値の分布を表わす多次元混合ガウス分布確率密度関数で定義します．

6.3　単語認識実験の概要

6.3.1　基本構成

発話された単語音声波形をスペクトル分析して得られた特徴時系列 $\boldsymbol{O}$ について，単語 HMM λ_i の Viterbi 確率 $P^*(\boldsymbol{O}|\lambda_i)$（式 5.13）を計算し，もっとも確率の高いモデル λ_i を選びます（図 6.3）．Viterbi 確率は $P(\boldsymbol{O}|\lambda_i)$ の近似値ですが，

Forward あるいは Backward 確率の計算に比べて計算量が少なく，実用上問題ないことが知られています．このため，実習では Viterbi アルゴリズムによって単語音声の認識を行っています．実用システムでは，誤った発声による誤動作を防

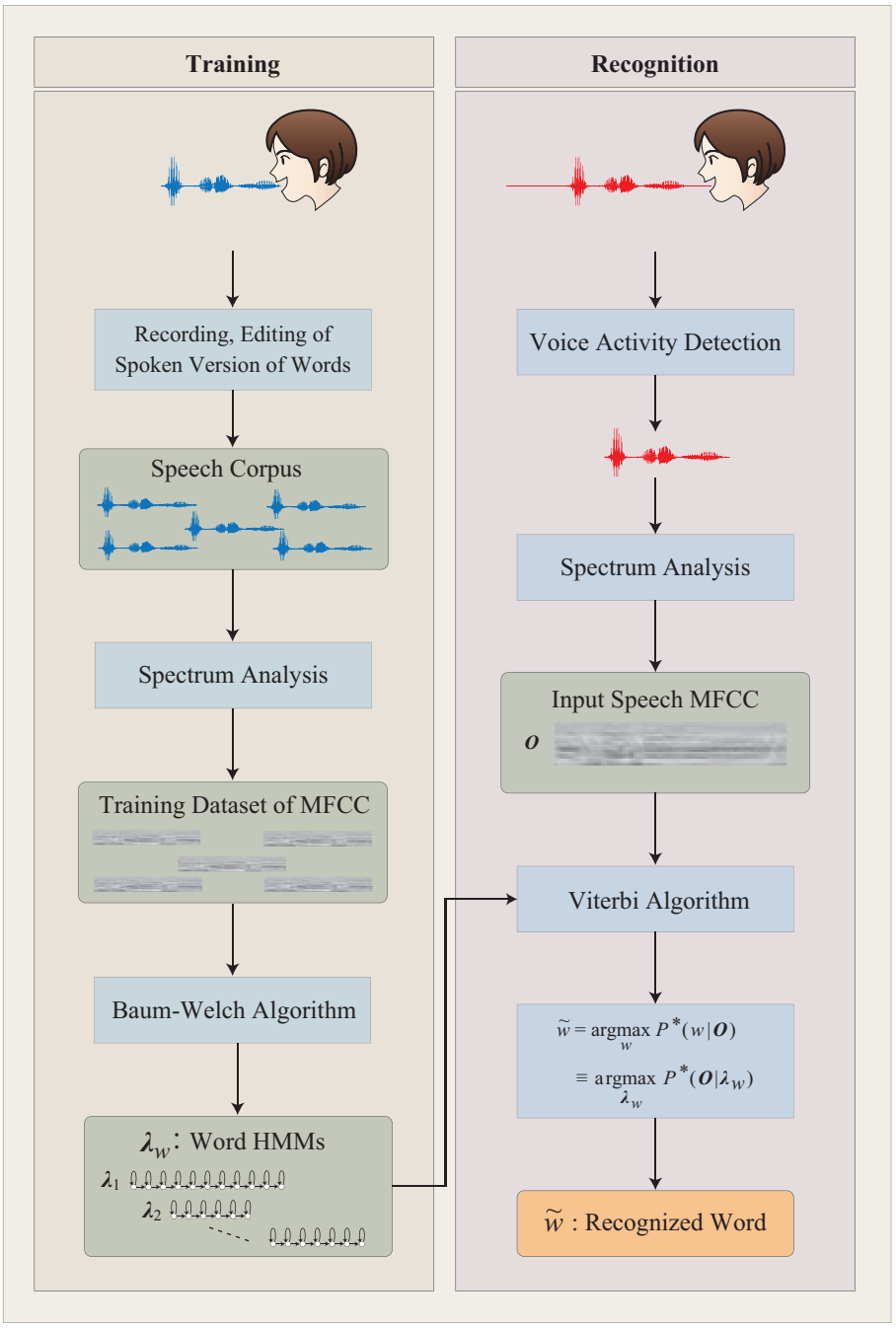


図 6.3: 単語認識実験の概要

ぐため，どの単語モデルも十分に高い確率値を出さない場合は，入力音声はどの単語でもないと判断し，入力を棄却するようにすることがあります．しかし，本実験では，入力音声はあらかじめ登録した単語のいずれかであるとします．したがって，どんな変な音が入力されても，$P^*(\boldsymbol{O}|\lambda_i)$ を計算することができれば[2]，認識対象の単語のうち最も確率が高いもの $\underset{i}{\operatorname{argmax}}\, P^*(\boldsymbol{O}|\lambda_i)$ が認識結果として出力されます．

[2]極端に短い音声が入力された場合などは，計算に失敗します．

6.3.2 音声区間検出

入力信号には無音の部分や音声でない信号が混ざっていることが多いので，スペクトル分析の前に，音声区間を推定する処理が必要です．この実習の On-The-Fly 単語認識（6-25 ページ）では，入力信号には無音の部分と音声の部分の 2 種類が含まれているという前提で単語認識を行います．無音といっても，マイクから入る背景騒音，マイクや信号処理回路に起因する微かな雑音が含まれているので，信号のパワーは 0 にはなりません．しかし，音声が存在する区間に比べれば，パワーはかなり小さい値を示します．そこで，本実験では信号のパワー（対数パワー）の値に基づいて，入力信号から自動的に音声区間を切り出す処理，すなわち**音声区間検出（Voice Activity Detection: VAD）**を行っています．

この実習では，音声波形を 20 ms 幅のセグメント（区間）に区切ります．そして，セグメント毎にその対数パワー

$$E = 10 \log_{10} \frac{1}{N} \sum_{i=1}^{N} x_i^2$$

を計算します．この値は音声が始まる前の無音区間では小さいですが，音声区間の始まりが近づくと急に大きくなります（図 6.4）．そこで，先頭のセグメントから順に対数パワー E の値を調べて，その値が閾値 θ を初めて超えたセグメントを A とします．次に，セグメント A の後に E が θ を初めて下回った時の最初のセグメントを B とします．音声区間の途中でも，無声子音などの区間では対数パワーの値が閾値を下回ることがあります（図 6.4 の第 24〜27 セグメント）が，通常その区間の長さは比較的短いです．このような区間を無音声と判断しないために，セグメント A の後に E が θ を下回っても，その状態が 5 セグメント（100 ms）連続しなければ，音声区間が終わったとは判断しません．

セグメント A とセグメント B を音声区間の開始および終了セグメントとすると，音声の開始および終了部の無声子音等のパワーが低い音声部分が切り取られてしまう可能性があります．したがって，A の前 9 個目のセグメント C，および B の後 9 個目のセグメント D をそれぞれ音声区間の開始セグメント，終了セグメントとしています（図 6.4）．

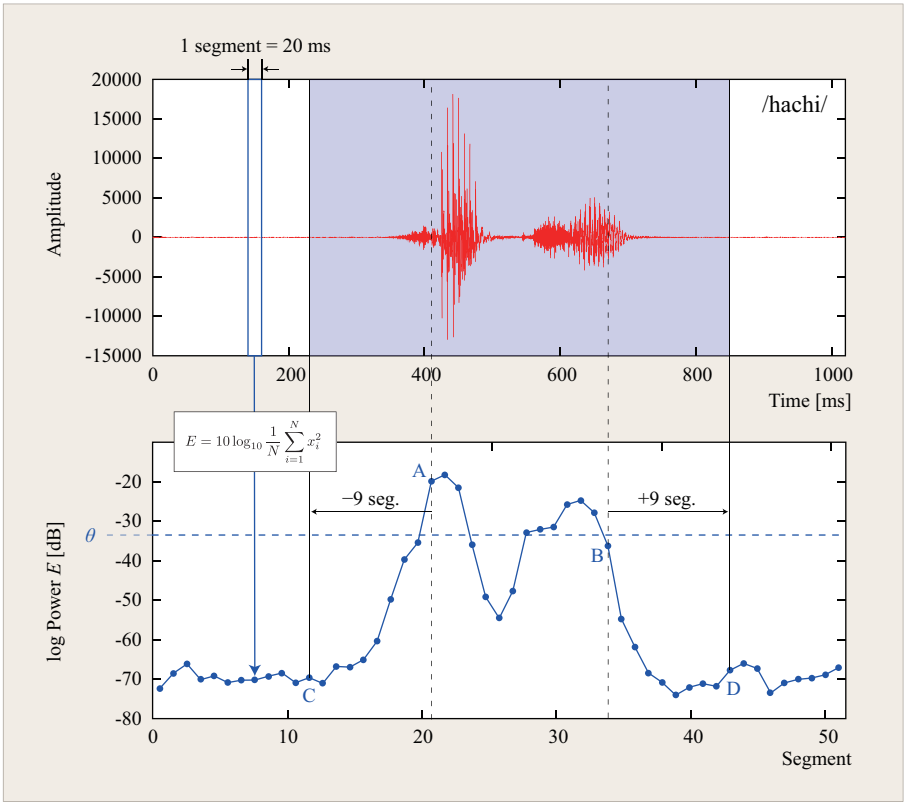


図 6.4: 音声区間検出の方式．上は音声波形（/hachi/）．下は対数パワーの変化グラフ．音声波形を 20 ms 幅のセグメント（区間）に区切り，セグメントの対数パワー $E = 10\log_{10}\frac{1}{N}\sum_{i=1}^{N} x_i^2$ を計算します．対数パワーの値が閾値 θ を初めて超えたセグメント A を検出します．A の後に閾値を初めて連続 5 セグメント下回った時の最初のセグメントを B とします．A の前 9 個目のセグメント C，および B の後 9 個目のセグメント D をそれぞれ音声区間の開始セグメント，終了セグメントとします．

この方法は，最も基本的な方法で，計算量が少ないという利点があります．しかし，音声波形の振幅は実行環境（録音レベル，マイクと口の距離など）に大きく影響されるので，実行環境毎に閾値の調整をする必要があります．雑音が多い（SNR が低い）環境では正しく音声区間の検出ができなくなります．

6.4 単語 HMM の学習

単語 HMM に用いる一方向性 HMM（5.4.1 節）の場合，初期状態確率は $\pi_1 = 1, \pi_{i>1} = 0$ に決まっているので，学習するパラメータは状態遷移確率 $\boldsymbol{A}$ とシンボル出力確率 $\boldsymbol{B}$ ということになります．状態数は単語の長さに応じて定めます．すなわち，短い単語に対しては状態数を少なく，長い単語に対しては状態数を多く定めます．

一方向性 HMM の学習を行うためには複数の学習用パターンが必要です．そこで，実習では多数の単語音声を録音し，スペクトル分析して MFCC パターンに変換します．そして，この多数の MFCC パターンを用いて，Baum-Welch アルゴリズム（5.3.4 節）により $\boldsymbol{A}$ と $\boldsymbol{B}$ を推定します．

Baum-Welch アルゴリズムにおいて，$\log P(\boldsymbol{O}|\tilde{\lambda})$ は単調増加します．その様子をグラフに描くと，例えば図 6.5 のようになります．$\log P(\boldsymbol{O}|\tilde{\lambda})$ の値は，学習の初期段階で急速に増加し，そのうち再推定を繰り返してもあまり変化しなくなることがわかります．この時点で学習が終わったと判断し学習を終了しています．

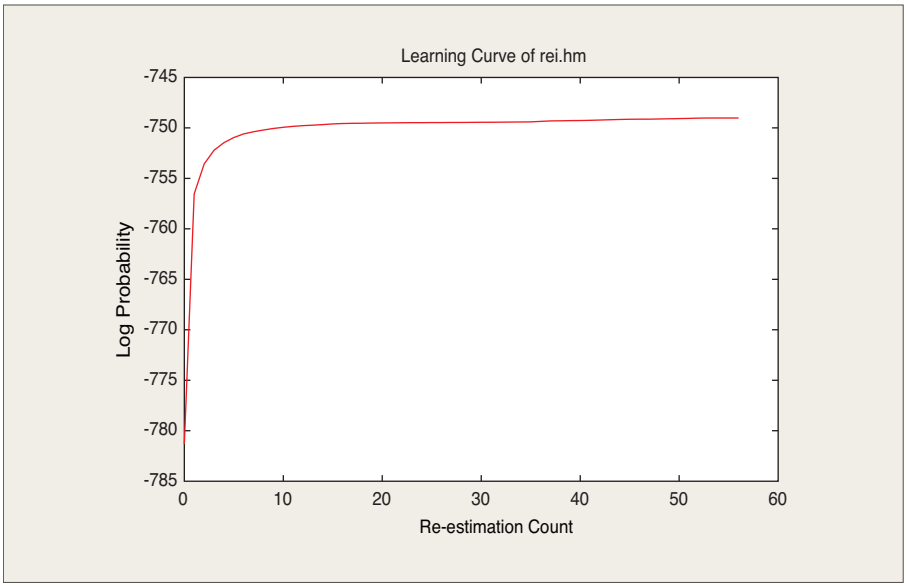


図 6.5: Baum-Welch アルゴリズムによる HMM の学習曲線．数字 0 の 50 個の学習用データ（`~takagi/mfcc/m0/rei_??.mfcc`）を用いて学習した場合．

6.5 実習

IED の端末にヘッドセットを接続し，自分の声で発声した単語の音声データを収録し，スペクトル分析を行って，単語 HMM の学習を行います．学習した単語 HMM を用いて単語認識を行います．収録した単語の音声データを用いて認識性能の評価も行います．

可能であれば、個性や創意を発揮し、認識対象単語（5 種類以上）はこのテキストの例と異なるものを考えてください．ただし，極端に短い単語や長い単語，促音「っ」を含む単語があると，この実習の On-The-Fly 音声認識では失敗する可能性が高いので、そのような単語は避けてください．

0〜9 の数字の認識を行うという設定で実習手順を説明します．音声データは`~/asr/wrecog/wav/digits.wav`，ラベルは`~/asr/wrecog/wav/digits.lab`として説明を進めます[3]．単語に複数の読み方がある場合は，その読み方の数だけ単語 HMM を作った方が良い結果が得られますが，ここでは簡単のため表 6.1 の 10 種類の発音とします．

表 6.1: 数字単語一覧の例

単語表記	単語名	発音	音素数	ファイル名
0	rei	/rei/	3	`rei_*`
1	ichi	/ichi/	3	`ichi_*`
2	ni	/ni/	2	`ni_*`
3	saN	/saN/	3	`saN_*`
4	yoN	/yoN/	3	`yoN_*`
5	go	/go/	2	`go_*`
6	roku	/roku/	4	`roku_*`
7	nana	/nana/	4	`nana_*`
8	hachi	/hachi/	4	`hachi_*`
9	kyuu	/kyuu/	3	`kyuu_*`

HMM の学習を行うためには，同じ単語の多数の発声が必要です．1 つの単語について 10 個の学習用音声データを用意してください．以下の説明は，この実習を`~/asr/wrecog` で行うことを前提としています．WaveSurfer はこのディレクトリに移動してコマンドラインで起動してください．

[3]テキストの実例では数字（digits）の音声データなので，このようなファイル名にしました．各自の認識対象に応じた内容を表わすファイル名を付けてください．適切なファイル名を付けるのは重要なことです．

6.5.1 認識用設定ファイルの作成

単語認識を行う場合，認識対象の単語の一覧表（表 6.1）に対応した設定アイルを用意する必要があります．認識タスクに応じて，予め用意されているファイルの内容を変更し，次の 2 種類のテキスト形式のファイルを作成してください．ファイル名は変えないでください．

単語 HMM 一覧 ファイル名＝`~/asr/wrecog/lib/HMMList`．表示文字列、単語名、HMM ファイル名の対応表です．3 カラムからなります．各カラムの区切りには半角スペースまたはタブを用います．空行があってはなりません．最後の行末には必ず改行を入れてください．

- 第 1 カラム：端末に表示される文字列（64byte まで；半角全角 OK）．
- 第 2 カラム：単語名（半角英数字 `[a-zA-Z0-9]`）．
- 第 3 カラム：HMM ファイル名（`~/asr/wrecog` 基準の相対パス名）．

```
0 rei   hmm/rei.hmm
1 ichi  hmm/ichi.hmm
2 ni    hmm/ni.hmm
3 saN   hmm/saN.hmm
4 yoN   hmm/yoN.hmm
5 go    hmm/go.hmm
6 roku  hmm/roku.hmm
7 nana  hmm/nana.hmm
8 hachi hmm/hachi.hmm
9 kyuu  hmm/kyuu.hmm
```

単語名一覧表 ファイル名＝`~/asr/wrecog/lib/wordlist`．単語名の一覧表

```
rei
ichi
ni
saN
yoN
go
roku
nana
hachi
kyuu
```

単語認識プログラム（recogf，countCorr，recog）には必要ありませんが，単語 HMM の学習曲線を描くための gnuplot スクリプト ~/asr/wrecog/drawLC（以下）は，単語名を自分の描く学習曲線の単語名に変更してください．

```
unset key
set xlabel "Re-estimation Count"
set ylabel "Log Probability"

# 以下 2 行の "rei" を自分の描く学習曲線の単語名に変更する．
set title  "Learning Curve of rei.hmm"
plot "log/rei.log" using 3 with lines

pause -1
set terminal png
set out "learningCurve.png"
replot
```

6.5.2 単語音声データの作成

多くの単語発話を収録する作業は，話者に負担がかかります．一度の発話で良いデータが取れることはないので，練習も兼ねて必要な発話数よりも多く発音して，その中から良いものを選びます．この実習での「良いデータ」とは，必ずしもアナウンサーのようなハッキリしていて綺麗な発音の音声ではなく，音声認識で用いる実際の発話に近いデータのことをいいます．

1. 録音
   (a) 収録の準備をします（3.10.3 節）．
   (b) 「設定」の「サウンド」パネルはデスクトップ画面に出したままにしておいてください．「サウンド」設定パネルが開いだ状態でないと，音声が入力されないことがあります．入出力音量は実験の途中で適宜調節するとよいでしょう．
   (c) 単語を 12 回繰り返して発話してください．ただし，学習に用いるのはこのうちの 10 個です．一般に発話が安定しない傾向が強い最初と最後の発話を除外するのが良いと言われています．単語の間は 1 秒程度空けてください．音が最大値を越えて割れないように音量に気をつけてください．一度に全ての単語発声データ（単語種類数× 12 回）を続けて録音してください．全ての発話の録音が終了したら，適当な名前を付けて（このテキストではファイル名を digits.wav として説明します．）~/asr/wrecog/wav ディレクトリに音声データを WAV 形式で

保存してください（保存のしかたの例は図 3.21）．保存したら，一旦このバッファを閉じてください（バッファ右端の「×」印をクリック．）

(d) 音声データの先頭から末尾に向かってラベルを入力していきます．

i. 保存した音声ファイルを開くと「Choose Configuration」パネルが現われるので，”HTK trainscription”を選んで「OK」ボタンを押します（図 6.6）．これで，ラベルが入力できるモードになりました．

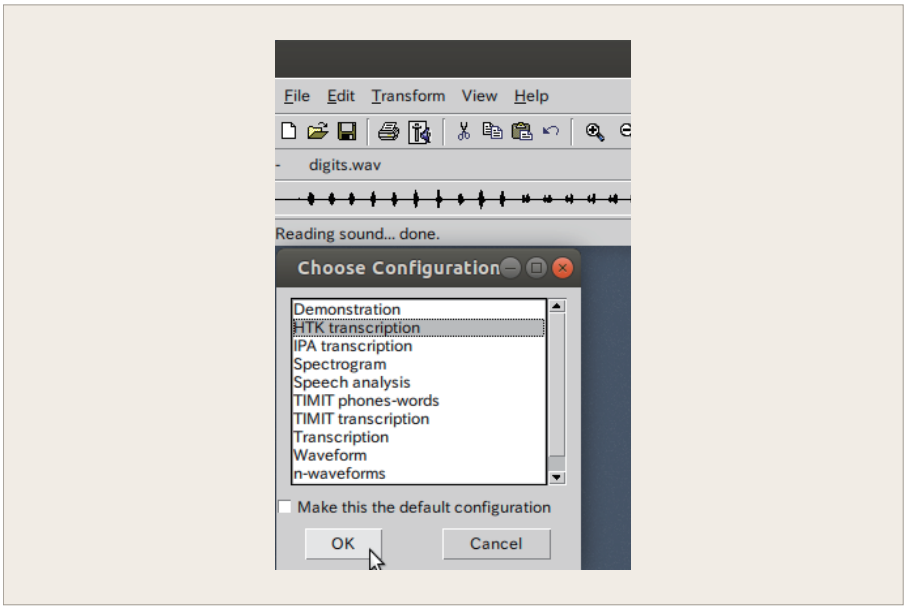


図 6.6: ラベルを入力するために”HTK trainscription” モードにする．

ii. ラベル形式の設定をします．ラベル入力領域（時間目盛「time」の下に「.lab」と表示されている初期状態で空白の欄）で右クリックして現われたメニューから「Properties...」を選択し，「Trans1」の「Label file format:」を”WaveSurfer” に設定して「Apply」ボタンを押してください（図 6.7）．この設定によりラベルファイルに記録される時間が秒単位になります．

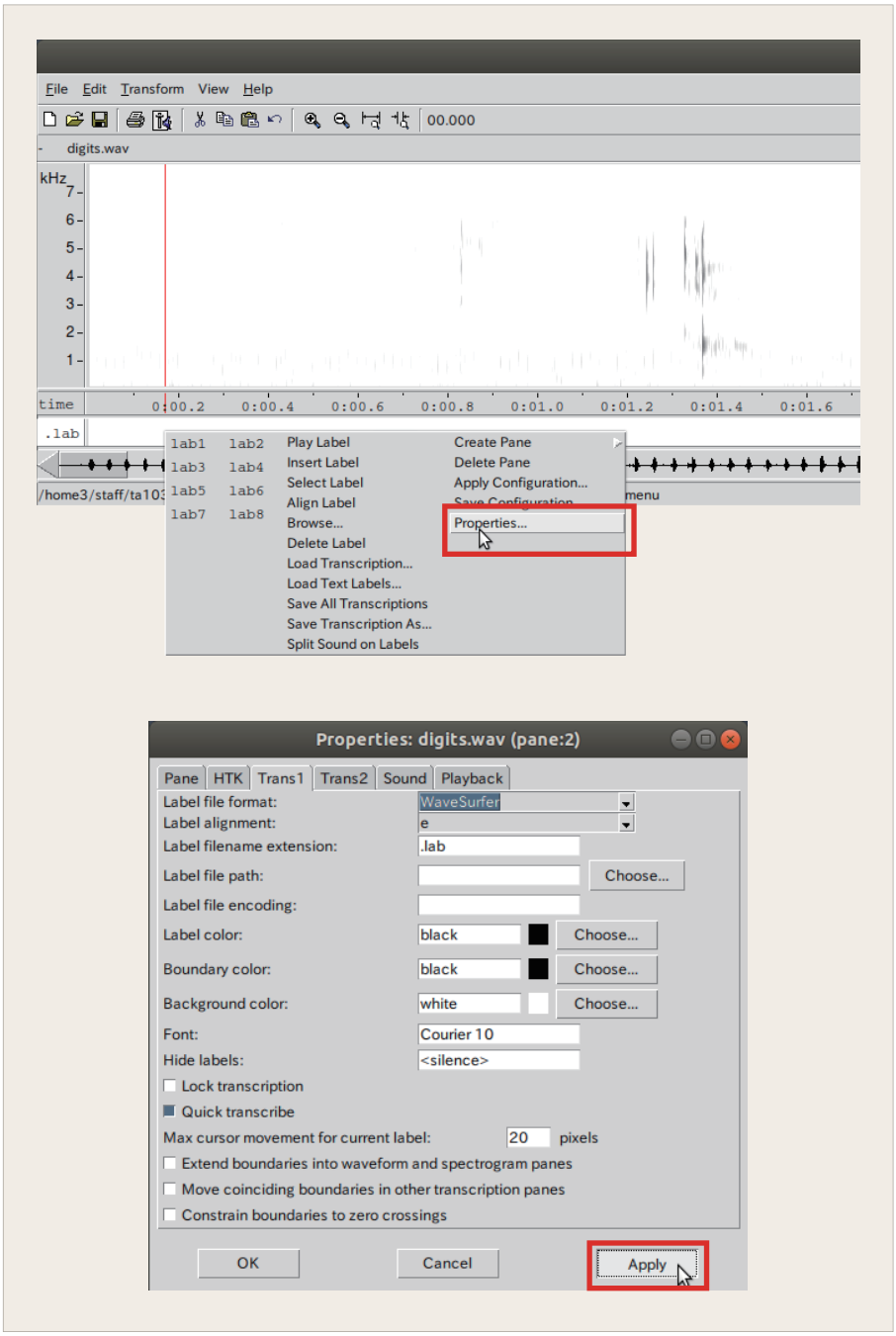


図 6.7: ラベル入力領域で右クリックし，「Properties...」を選択し，「Trans1」の「Label file format:」を”WaveSurfer” に設定して「Apply」ボタンを押す．つぎに「OK」ボタンを押してパネルを閉じる．

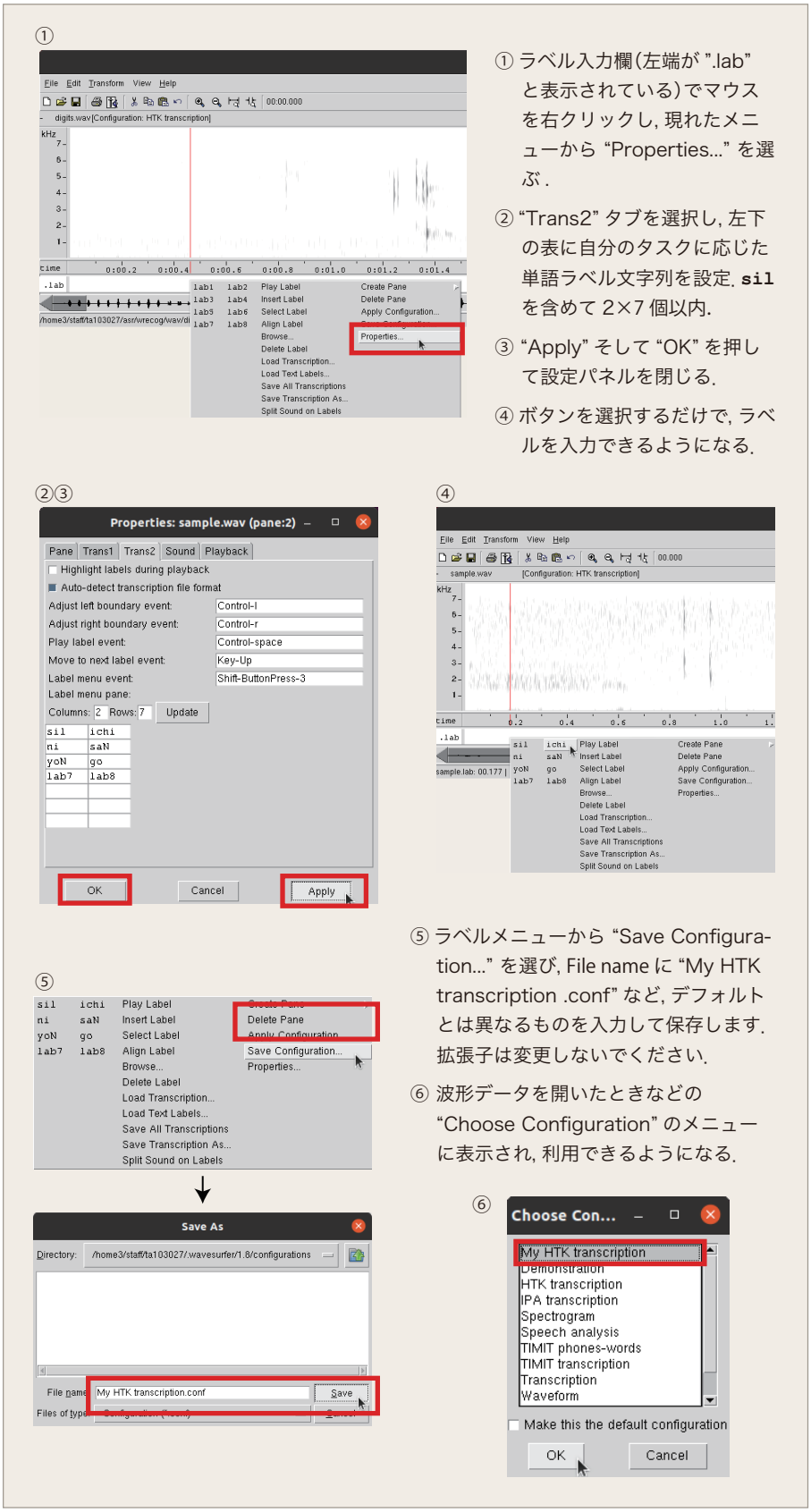


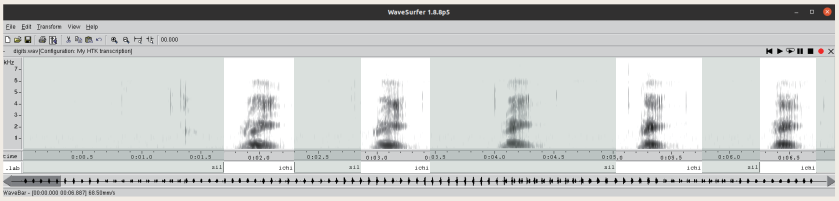
図 6.8: 自分のタスクの単語一覧にあるラベルを設定する.

iii. 単語ラベル文字列を WaveSurfer のラベルに設定します (図 6.8).
まず, ラベル入力欄（左端が".lab"と表示されている）でマウス

を右クリックし，現れたメニューから”Properties...”を選んでください．次に，”Trans2” タブを選択し，左下の表に自分のタスクに応じた単語ラベル文字列をキーボードから入力してください．音声（あるいは学習に使わない区間）を表す`"sil"`というラベルは必ず設定してください．そして”Apply”と”OK”を押して設定パネルを閉じます．これで，ボタンを選択するだけでラベルを入力できるようになりました．最後に，ラベルメニューから”Save Configuration...”を選び，File name に，たとえば，”My HTK transcription.conf” など，デフォルトとは異なる名前を指定して入力したラベルを保存します．拡張子は変更しないでください．この設定は次に波形データを開いたときなどの”Choose Configuration”のメニューに表示され，利用できるようになります．WaveSurfer のラベルは 2×7 が最大になっていて，これを超える数の設定はできません．バグと推測されますが，回避方法が見つかっていません．`"sil"`を含めて 14 個以内にしてください．ちなみに，ラベル入力領域に任意の英数字文字列をキーボードから直接入力することもできます．

iv. HMM の学習に用いる発話にラベルを付けます．単語の前後に 100ms（0.1s）の余白を付けてください．横軸の表示倍率によって目盛の時間幅が異なるので，間違えないように十分注意してください．

v. スペクトログラムを見たり，音声の一部を選択して（図 3.19）再生したりして，単語音声の始端と終端の位置を調べます．この実習で音声認識タスクの実例として説明に用いている数字音声の音響的特徴については付録 B で説明しています．ラベル付けの参考にしてください．音声ファイルの時刻 0 秒から最初の単語ラベルの開始時間（単語開始時間の 100ms 前）までの区間には`"sil"`というラベルを付けます．音が入っていても学習に使いたくない区間にも`"sil"`を付けます（図 6.9）．



時刻 0 秒から最初の学習データまでは ”**sil**” のラベルを付ける. 学習データの区間には発話内容に応じたラベルを付ける. 学習に使わない区間には ”**sil**” のラベルを付ける.

図 6.9: 時刻 0 秒から最初の学習データまでは`"sil"`のラベルを付ける. 学習データの区間には発話内容に応じたラベルを付ける. 学習に使わない区間には`"sil"`のラベルを付ける.

vi. まず、ラベル入力領域において単語の始端から 100ms 前の位置にカーソルを移動して右クリックします．ポップアップメニューから入力したいラベルを選んでください（図 6.10）．ラベル情報には音声区間の開始時刻と終了時刻が必要ですが，最初のラベル挿入操作時はその境界で終了する音声区間の開始時刻が入力されていないので、WaveSurfer からエラー表示されます（図 6.11）．その音声区間の開始時刻が指定されていないという意味です．これは最初のラベルを入力するときだけであり，この後の作業への影響は無いので，「Skip Messages」を押して無視してください．

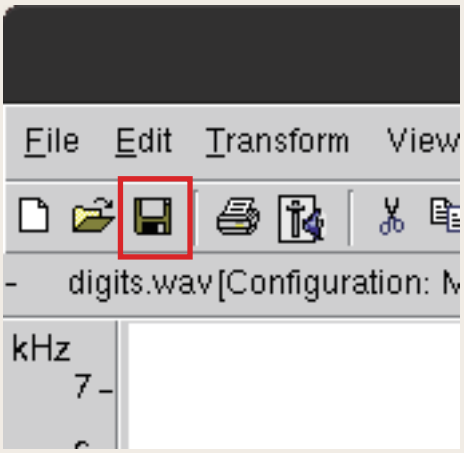
vii. この要領で，終端から 100ms 後ろの位置にカーソルを合わせ，全てのラベルを入力します．学習データに不適当と思われるデータは”sil”の範囲に含めます．

区間の変更 ラベルを入力した後でも区間の境界を移動させて調節することができます．

ラベルの音声再生 ラベル区間内で右クリックしたときに表示されるメニューで「Play Label」を選択すると，この区間の音声が再生されます．この機能を用いて，開始と終端が正しいかどうか確認することを推奨します．

ラベルの削除 ラベル区間内で右クリックしたときに表示されるメニューで「Delete Label」を選択します．

(e) ラベルの保存は，ファイル保存アイコン (図 6.12) をクリックすることによって行います. 作業途中で中断しても，保存しておけば後で再開して作業を続けることができます．



① ラベルの保存は, ファイル保存アイコン(左図赤枠)をクリックすることによって行う.

② 音声波形データのファイルとは別のファイルが自動的に生成され, そのファイルにテキスト形式で保存される.

・ラベルファイル名は音声波形データファイルの拡張子を **wav** から **lab** に変えたもの.

・既にラベルファイルがある場合は, 上書きされる. 上書きされる場合に警告ダイアログは表示されない.

図 6.12: ファイル保存アイコンをクリックして入力済みラベル情報を保存.

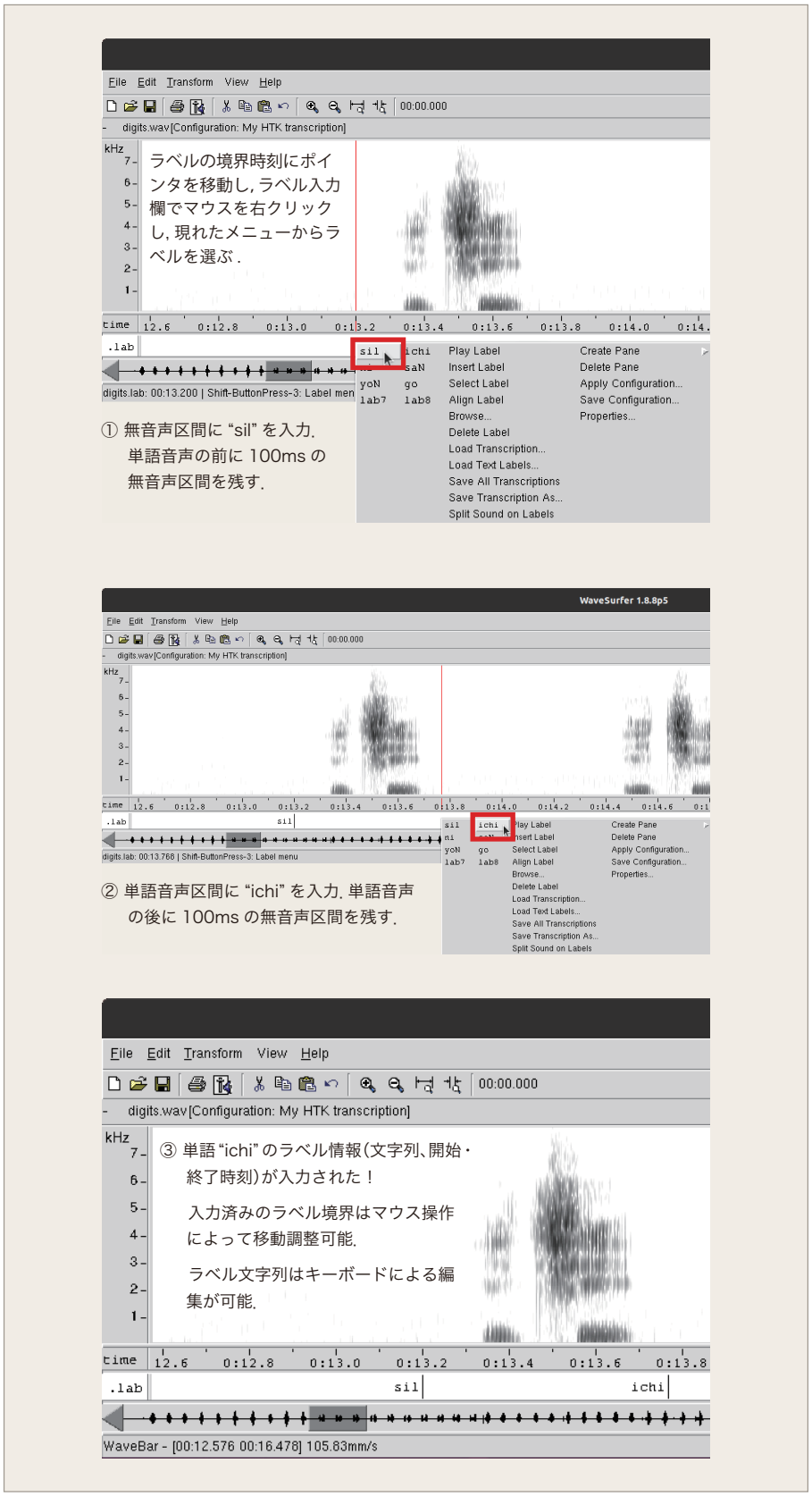


図 6.10: ラベル入力の実施例．境界時刻位置のラベル入力欄で右クリックし，メニューのラベルを選ぶと，この位置を音声区間の終端とするラベルが挿入される．入力済みラベルの境界や文字列はマウスやキーボードの操作で変更可能．

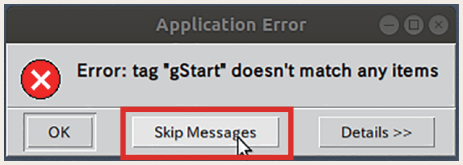


図 6.11: ラベル入力時に WaveSurfer からエラーが表示されることがあるが「Skip Messages」を押して無視し，やり直せばよい．

2. ラベルデータの検査

   ラベルデータはテキスト形式で記録されているので，端末に表示したり（図 6.13），テキストエディタで編集することができます．ラベルが正しく付与されているかどうか検査してください．

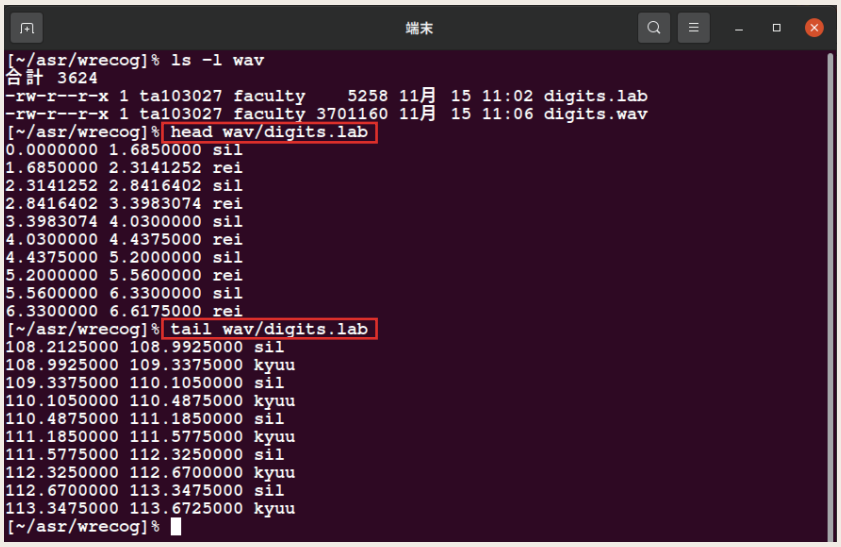


図 6.13: WaveSurfer で付けた HTK 形式の音声セグメントラベル．ラベルファイルの拡張子は”lab’．時間数値の単位は秒．

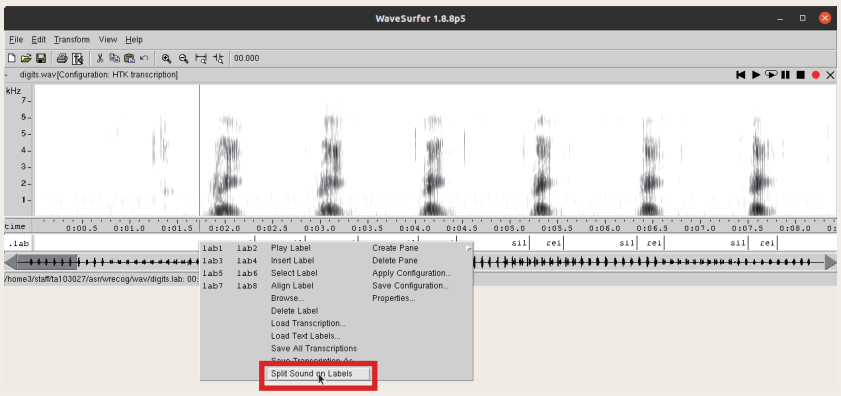
次の点に注意します．

時間の数値が秒単位として妥当であるか？ ラベルの入力形式が適切でない場合は，大きな整数値になっていることがあります．WaveSurfer で当該データを読み込み，ラベル形式を設定し直した（前述）後，ラベルファイルを保存し直すことによって時間が秒単位になります．

単語名（“rei”，”ichi” など）が正しく揃っているか？ 綴りが 1 文字でも異なっていると，異なる単語データとして処理されてしまいます．必要があれば，エディタで編集して整えてください．

余分なラベルが記録されていないか？ ラベル入力作業の過程での手違いにより意図しないラベルが入力されてしまっていることがあります．必要があれば，エディタで編集して整えてください．

上記の修正をした場合，WaveSurfer で当該データを読み込んで，音声波形とラベルの対応を確認してください．



ラベル入力欄（左端が ".lab" と表示されている）でマウスを右クリックし，
現れたメニューから "Split Sound on Labels" を選ぶと，
ラベル情報を用いて音声波形データが切り分けられ，ファイルに保存される．

図 6.14: WaveSurfer のラベル入力欄（左端が".lab"と表示されている）でマウスを右クリックし，現れたメニューから"Split Sound on Labels"を選ぶと，ラベル情報を用いて音声波形データが切り分けられ，ファイルに保存される．

3. **単語音声学習データの作成**

WaveSurfer で付けたラベルに基づいて，音声波形データから個々の単語の発話を切り取り，別々のファイル保存します．切り取られた音声波形の小区間をセグメント（segment）といいます．WaveSurfer のラベル入力欄（左端が`.lab`と表示されている）でマウスの右クリックし，現れたメニューから "Split Sound on Labels" を選ぶと（図 6.14），ラベル情報を用いてセグメントデータが生成されてファイルに保存されます．音声波形ファイルが`digits.wav`，そのラベルファイルが`digits.lab`である場合，音声波形ファイルとラベルファイルが格納されているディレクトリの下に`digits.wav.split`という名前のサブディレクトリが作られ、その中にセグメント波形データが保存されます．

どんなセグメント波形ファイルができたのか、確認してみましょう．このテキストの説明のための例の場合，次のような 6 桁のセグメント番号，ラベル文字列，拡張子（`wav`）からなる WAV ファイルが生成されていました．

```
[~/asr/wrecog]% ls wav/digits.wav.split/ | head
000000rei.wav
000001sil.wav
000002rei.wav
000003sil.wav
000004rei.wav
000005sil.wav
000006rei.wav
000007sil.wav
000008rei.wav
```

```
000009sil.wav
[~/asr/wrecog]% ls wav/digits.wav.split/ | tail
000189sil.wav
000190kyuu.wav
000191sil.wav
000192kyuu.wav
000193sil.wav
000194kyuu.wav
000195sil.wav
000196kyuu.wav
000197sil.wav
000198kyuu.wav
```

`ls` コマンドで保存先として指定したディレクトリの内容を表示し，ファイルを確認してください．学習データの準備はとても重要です．念のため，ファイル名一覧だけではなく，ファイルサイズもチェックしておきましょう．例えば，単語の長さが 0.5 秒の場合，その WAV ファイルのサイズ（バイト数）はいくつですか？ ちなみに，この実験の音声データ形式（16kHz サンプリング，1 チャネル，liner PCM）に対する WAV データのヘッダサイズは 44 バイトです．

4. 作成した単語音声の検聴

   念のため，単語音声データが正しく作成されたかチェックします．作成した音声波形ファイルをひとつひとつ検聴してください．例えば`"rei"`とラベル付けされた音声セグメントを一括検聴する場合は、端末で，

```
[~/asr/wrecog]% foreach f (`ls wav/digits.wav.split/*rei.wav`)
foreach? echo $f
foreach? play $f
foreach? end
```

   のように入力すると，ワイルドカードに一致する音声ファイルについて，順番にファイル名が表示され音声が再生されます．検聴の結果によって，必要があれば，音声を収録し直したり、ラベルを付け直すなどしてください．

   `foreach` 文の括弧の中の ` はバッククオート（back quote）であることに注意してください．IED 教室の端末のキーボードでは「`@`」のキーに割り当てられています（図 6.15）．このテキストの PDF 版からコピー＆ペーストで端末にコマンドを入力すると，表示フォントの影響でバッククオートが正しく入力されません．バッククオートの部分は直接キーボードから入力してください．

音声データを再生する `play` コマンドを実行した際に表示される “play WARN

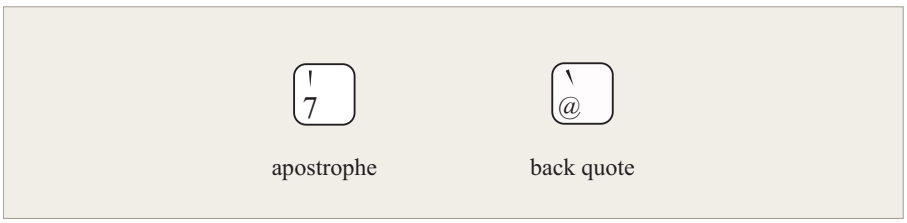


図 6.15: apostrophe と back quote の IED 教室端末のキー位置.

alsa: can't encode 0-bit Unknown or not applicable" のメッセージは無視して結構です.

6.5.3 スペクトル分析

録音した全ての音声波形をスペクトル分析して MFCC に変換し，ファイルに保存します．スペクトル分析には第 3 章で作成した `mfccf` を用います．個々の音声ファイルごとにコマンドを入力するのは大変なので，シェルの機能を利用し，以下のように一括処理を行います．

```
[~/asr/wrecog]% foreach f ( `ls wav/digits.wav.split/*.wav` )
foreach? set g = $f:t:r
foreach? set h = mfcc/${g}.mfcc
foreach? mfccf $f mfcc/${g}.mfcc
foreach? end
```

ここで，`:t`(tail)はファイル名からパス名を取り除いた部分を取り出す，`:r`(root)はファイル名から拡張子を取り除いた部分を取り出す作用をします．MFCC の値を保存したファイルは，`mfcc` というディレクトリに保存されます．MFCC のファイルの数は必ず確認しておいてください．セグメントの種類毎の数（この例では `rei`）を確認するためには以下のようにします．`-l` の `l` は数字の 1 ではなく，エル ℓ です．

```
[~/asr/wrecog]% ls mfcc/??????rei.mfcc | wc -l
10
```

6.5.4 単語 HMM の学習

1. 単語 HMM の学習用データの一覧を単語毎に作成します．データの一覧はディレクトリ `~/asr/wrecog/lib` に保存してください．たとえば，単語「0」の場合，

   ```
   [~/asr/wrecog]% ls mfcc/??????rei.mfcc > lib/rei.list
   ```

   と入力します．実行後に `lib/rei.list` の内容を確認しておいてください．この作業も `foreach` を使えば，一度に行うことができます．単語名の一覧を `lib/wordlist` というテキストファイルに用意しておけば，以下のように一括処理することができます．

```
[~/asr/wrecog]% foreach d ( `cat lib/wordlist` )
foreach? ls mfcc/??????${d}.mfcc > lib/${d}.list
foreach? end
```

2. Baum-Welch アルゴリズム（5.3.4 節）を用いて，単語 HMM を学習します．状態数は単語に含まれる音素の数に応じて決めます．この実習の実験条件においては、単語を構成する各音素と前後の無音声区間に 1 状態を割り当てるのが適当と思われます．音素（第 2.2 節）1 つあたり 1 状態を割り当ててください．長音（`/aa/`，`/ii/`，`/uu/`，`/ee/`，`/oo/`）や促音（`/Q/`）は 1 音素とみなします．第 2 章の言語の二重文節の説明図（図 2.1）では，言語的に長音を 2 音素としていますが，音声現象としては長音は短音の継続長が長くなったものと見做して差し支えありません．さらに，単語の開始前と終了後に無音声（`sil`）のための 1 状態を割り当ててください（図 6.16）．さらに，単語の開始前と終了後に無音声（`sil`）のための 1 状態を割り当ててください．したがって，例えば，「電気通信大学（`/deNkitsuushiNdaigaku/`）」という単語の HMM の状態数は 19 とします．このようにして決めた状態数で HMM の学習を試みますが，学習が失敗する場合は状態数を増減して試してください．本実習では学習データが少ないので，状態数を減らす方向に調整した方が学習が成功することが多いと思われます．ファイルには，必ず単語名（表 6.1）に対応した名前を付けてください．

単語（正書法）	音素＋無音声区間	HMM 状態数（本実験での目安）
太郎	sil t a r oo sil	6
は	sil w a sil	4
学校	sil g a Q k oo sil	7
へ	sil e sil	3
行った	sil i Q t a sil	6

図 6.16: 本実験の条件において単語 HMM に割り当てる状態数の目安．音素 1 つあたり 1 状態．長音（`/aa/`，`/ii/`，`/uu/`，`/ee/`，`/oo/`），促音（`/Q/`）は 1 音素とみなす．単語の開始前と終了後に無音声（`sil`）のための 1 状態を割り当てる．

```
[~/asr/wrecog]% train lib/rei.list 5 hmm/rei.hmm log/rei.log
```

と入力し，HMM の学習を実行します[4]．学習が終了するまでに数分かかる場合があります．lib/rei.list は学習データの一覧です．hmm/rei.hmm は学習された HMM のパラメータ値が保存されているテキストファイルです．パラメータの再推定毎の $\log P(\boldsymbol{O}|\tilde{\lambda})$ の値が log/rei.log に記録されます．どれか 1 つの HMM の $\log P(\boldsymbol{O}|\tilde{\lambda})$ の値を用いて，図 6.5 のような学習曲線のグラフを作ってください．この図は，gnuplot のバッチファイル drawLC を使い，

```
[~/asr/wrecog]% gnuplot drawLC
```

と入力して作りました[5]．上記を実行すると学習曲線が表示されます．表示内容を確認してリターンキーを押すと，learningCurve.png というファイルに図のデータが保存されます．

学習の失敗　学習プログラム train を実行すると，以下のように学習が失敗することがあります．

```
[~/asr/wrecog]% train lib/hachi.list 6 hmm/hachi.hmm log/hachi.log
training_data= lib/hachi.list
tokens= 10
frame= 556
dim= 20
state= 6
mix= 4
hmm_file= hmm/hachi.hmm
log_file= log/hachi.log
(0) logp= -1.011268e+03
(1) logp= -9.675306e+02
(2) logp= -9.605083e+02
(3) logp= -nan
```

この例のように，$\log P(\boldsymbol{O}|\tilde{\lambda})$ の値が途中で nan（非数）となり，ここで学習プログラムが終了してしまいます．学習に失敗する場合は, 次のような原因が考えられます．

- 音声波形データの準備が正しくなされていない．
- スペクトル分析に失敗している．

[4] この train という学習プログラムは HMM の混合数を 4 に固定しています．
[5] 認識させる単語セットによってファイル名は異なることになるので，drawLC は自分用に編集して使ってください．

- 音声が正しく収録されていない（単語の途中で切れている，音が割れている，など）
- 学習データに違う単語のデータが混ざっている
- 状態数が少なすぎる，または多すぎる

単純な学習作業なので，学習データが整っていれば，失敗することはほとんどありません．学習に失敗する場合は，音声収録から学習までの手順を見直してください．作業の過程で作るファイルの数，ファイル名，サイズ，ファイル形式，内容などをチェックするとよいでしょう．上記の失敗例の場合，学習データに問題は無かったので，状態数を 6 から 5 に変えてみたら，以下のように学習が成功しました．

```
[~/asr/wrecog]% train lib/hachi.list 5 hmm/hachi.hmm log/hachi.log
training_data= lib/hachi.list
tokens= 10
frame= 556
dim= 20
state= 5
mix= 4
hmm_file= hmm/hachi.hmm
log_file= log/hachi.log
(0) logp= -7.223173e+02
(1) logp= -6.907365e+02
(2) logp= -6.867466e+02
(3) logp= -6.852862e+02
...
(46) logp= -6.766227e+02
(47) logp= -6.766215e+02
(48) logp= -6.766205e+02
```

6.5.5 学習の検証

単語 HMM が学習できたので認識してみましょう．この実験には recogf という単語認識プログラムを用います．HMM の学習データの認識を行い，学習が出来ているかどうかの検証をします．まず，recogf というコマンドに，数字単語 HMM 一覧ファイル lib/HMMList と認識したい音声の MFCC ファイルを指定すると，ビタビ確率 $\underset{i}{\operatorname{argmax}}\, P^*(w_i|\boldsymbol{O})$，$w_i \in \mathcal{W}$（式 5.13）を計算して表示します．以下に一例を示します．

```
[~/asr/wrecog]% recogf lib/HMMList mfcc/000000rei.mfcc
0 rei
```

```
[~/asr/wrecog]% recogf lib/HMMList mfcc/000040ni.mfcc
2 ni
[~/asr/wrecog]% recogf lib/HMMList mfcc/000060saN.mfcc
3 saN
[~/asr/wrecog]% recogf lib/HMMList mfcc/000140nana.mfcc
7 nana
```

認識させたいファイルをいくつか指定してみましたが，ここで認識させたものについては全て正しく認識されました．

次に，学習に用いた単語音声を認識し，正しく認識されたデータの個数を数えてみます．認識対象となるのは `lib/wordlist` に書かれている単語です．

```
[~/asr/wrecog]% countCorr
Evaluation start.  Wait a moment.
rei, 10
ichi, 10
ni, 10
saN, 10
yoN, 10
go, 10
roku, 10
nana, 10
hachi, 10
kyuu, 10
```

10 個の識別で，かつ学習データを用いて評価しています．認識率は 100%となりました．HMM の性能評価としての意味は無く，HMM の学習の確認と次の On-The-Fly 認識に必要な各種設定ファイルが正しく作られていることの確認の意味があります．認識性能の評価をするためには，On-The-Fly 認識の他，学習に用いない性能評価用の音声データを用います．

6.5.6 On-The-Fly 単語認識

単語 HMM の学習に成功したら、いよいよマイクから入力した音声を直接認識してみましょう．認識処理の結果は端末に表示されるだけなので，適宜適当なファイルに保存しておいて，後で実行例としてレポートに使ってください．「設定」の「サウンド」パネルはデスクトップ画面に出したままにしておいてください．「サウンド」設定パネルが開いだ状態でないと，音声が入力されないことがあります．入出力音量は実験の途中で適宜調節するとよいでしょう．

音声区間検出の閾値の決定

音声認識では，マイクから入力される音響信号から音声が存在する区間を自動的に検出します．この実習では音量に基づいて音声区間を検出（6.3.2 節）しま

す．最初に**音声区間検出の閾値を決め**ます．ヘッドセットを装着します．vu というコマンドを用いると，マイク入力のパワーレベルの概算値が分かります．なにもしゃべらない時のレベル値より少し大きい値を音声区間検出の閾値の目安とするとよいでしょう．はじめに，端末エミュレータの横幅を少し広げておいてください．一行の文字数が不足すると，表示が "流れて" しまいます．

表示は状況に応じて変化します．下記の例は，ある時点の状況を示したものです．グラフ上の「*」は最大値の位置を示しています．この状況の場合，−25 dB に定めるのが適当だと思われます．終了するためには，Ctrl+C をキー入力します．

```
[~/asr/wrecog]% vu

 Short Time Speech Power
-60     -50       -40      -30      -20      -10       0 dB
 +-------+-------+-------+-------+-------+-------+
rec WARN alsa: can't encode 0-bit Unknown or not applicable
.==========================              *
```

この際に表示される "rec WARN alsa: can't encode 0-bit Unknown or not applicable" のメッセージは無視して結構です．

単語音声認識の実行

閾値を決めたら，次は，**単語認識のプログラムを起動**します．音声認識を行うためには，例えば以下のように端末で recog コマンドを起動します．

```
[~/asr/wrecog]% recog -25 lib/HMMList
```

"−25" は vu の表示を見て決めた音声区間検出の閾値 [dB] です．この数字は一例です．大きい値を指定すると，音声パワーの大きい部分だけが処理されるようになり，音声の最初と末尾が切れる可能性が高くなります．値を小さくすると，音声の最初と末尾が切れる可能性は減りますが，音声以外の音も処理される可能性が高まります．認識がうまく行かない場合は，適宜調整してみてください．

「Return キーを押してください」と表示されます．この状態は音声の入力を行っていませんので，話しても処理しません．Return キーを押して，単語を発声してください．認識処理が終わると，端末結果が表示されます．単語の順位，単語表記，単語名，尤度が，尤度（$\log P^*(\boldsymbol{O}|\lambda)$）の降順に表示されます．実際に入力した単語が第 1 位の単語と一致していれば認識成功ということになります．音声区間検出によって検出された音声波形は~/asr/wrecog/vadwav に保存されます．発話毎に異なったファイル名が付けられます[6]．On-The-Fly 単語認識を終了するためには，Ctrl+C をキー入力します．この際に表示される "rec WARN alsa: can't encode 0-bit Unknown or not applicable" のメッセージは無視して結構です．

[6]MFCC 分析を行ったプロセスのプロセス番号で名前を振っています．

```
[~/asr/wrecog]% recog -25 lib/HMMList
Return キーを押してください
Isolated Word Recogition On-The-Fly
rec WARN alsa: can't encode 0-bit Unknown or not applicable
vad_file= vadwav/76302.wav
--------------------
rank word (kana)= log-likelihood
--------------------
 1. 8 (hachi)= -952.510926
 2. 1 (ichi)= -977.724834
 3. 3 (saN)= -1032.404311
 4. 0 (rei)= -1087.581653
 5. 2 (ni)= -1167.379263
 6. 4 (yoN)= -1227.270401
 7. 6 (roku)= -1240.591021
 8. 7 (nana)= -1261.096937
 9. 5 (go)= -1360.261797
10. 9 (kyuu)= -1373.558098

ハングアップ
Return キーを押してください
Isolated Word Recogition On-The-Fly
rec WARN alsa: can't encode 0-bit Unknown or not applicable
vad_file= vadwav/76327.wav
--------------------
rank word (kana)= log-likelihood
--------------------
 1. 8 (hachi)= -911.710779
 2. 1 (ichi)= -916.062055
 3. 0 (rei)= -1009.450527
 4. 3 (saN)= -1066.173722
 5. 2 (ni)= -1104.335671
 6. 4 (yoN)= -1208.086255
 7. 6 (roku)= -1217.270178
 8. 7 (nana)= -1236.537874
 9. 5 (go)= -1378.731672
10. 9 (kyuu)= -1437.539677
ハングアップ
Return キーを押してください
Isolated Word Recogition On-The-Fly
rec WARN alsa: can't encode 0-bit Unknown or not applicable
```

```
vad_file= vadwav/76349.wav
--------------------
rank word (kana)= log-likelihood
--------------------
 1. 0 (rei)= -nan
 2. 1 (ichi)= -nan
 3. 2 (ni)= -nan
 4. 3 (saN)= -nan
 5. 4 (yoN)= -nan
 6. 5 (go)= -nan
 7. 6 (roku)= -nan
 8. 7 (nana)= -nan
 9. 8 (hachi)= -nan
10. 9 (kyuu)= -nan
```

認識結果の検討

この実行例では，1 回目と 2 回目の音声入力では，第 1 位の単語が発話した単語に一致していました．しかし，3 回目の音声入力に対しては単語 HMM の尤度の計算に失敗しました．音声区間検出によって切り出されて尤度計算に用いた音声波形は vadwav/76349.wav （認識結果の上に表示）に保存されています．認識に成功した第 2 回目の音声波形 vadwav/76327.wav と比べてみましょう（図 6.17）．

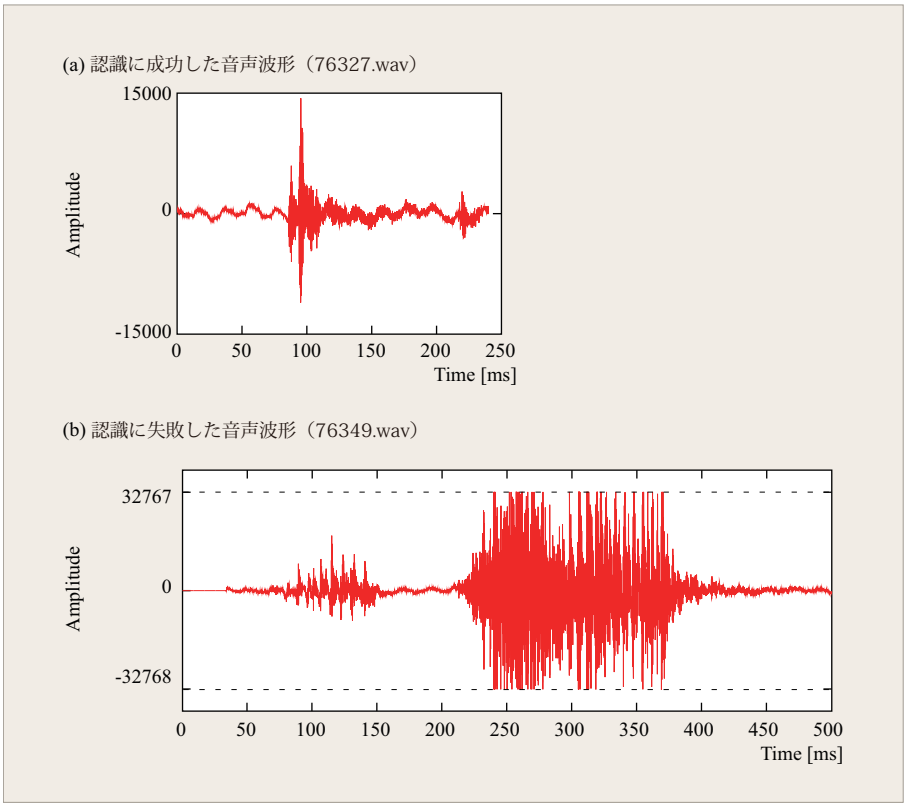


図 6.17: On-The-Fly 単語認識に (a) 成功した音声と (b) 失敗した音声の例．(b) の場合は入力レベルが大きすぎて符号付 16 ビット整数の値の範囲を超えてしまっている．この図は `ad2txt16` の出力を gnuplot で表示したものを説明用に修整したもの．

音声波形の振幅は符号付 16 ビット整数で表現されています．すなわち，値の範囲は −32768〜+32767 です．認識に成功した音声波形（図 6.17(a)）は振幅値がこの範囲に収まっていますが，失敗した音声波形（図 6.17(b)）は振幅値が超えてしまっている部分があります．声が大きすぎたためです．

認識結果が間違っていると分かって声を張って発話すると，このような入力となり，かえって失敗することがあります．落ち着いて，単語 HMM の学習用の音声データを発話したときと同じ程度の音量で発話しましょう．

その他に、認識が失敗する事例で良くあるのは，入力された音声が途中で切られてしまう場合です．音声区間自動検出の閾値に比べて単語音声のパワーが小さすぎる場合，単語の途中の破裂子音（/k/, /t/, /g/, /d/, /p/）の閉鎖部（図 2.12）で単語が終わったと判断され，ここで切られた音声が認識処理されることがあります．

On-The-Fly 単語認識で正解が得られるための条件は以下の通りです．

1. 認識対象の単語が入力される．
2. 学習データと同じ話者が発話する．
3. 学習データと同じ発音で発話する．
4. 学習データと同じ条件（マイク等の種類，マイクの位置，背景雑音の状態

など）で入力される．

5. 音声区間検出の閾値が適切である．

6. 適切な音量で入力され，音声区間検出が正しく実行される．

7. 認識対象の単語に類似した発音の単語が無い．

満たされない条件があると，単語認識が失敗する可能性が高まります．認識に失敗した場合は，その原因について考察してください．そして，原因と推定される条件を改善してやり直してみましょう．

WaveSurfer で vadwav に保存されている音声波形を表示したり，聞いてみたりすると，認識の成否の原因を知ることができる場合が多いです．是非，調べてみてください．

6.5.7 プログラム構成

この実習では、最も単純な音声認識タスクである**特定話者孤立単語音声認識（Speaker-Dependent Isolated Spoken Word Recognition）**を行いました．**特定話者（Speaker-Dependent: SD）**とは，ある一人の話者の音声のみに対応しているということです．**孤立単語音声認識（Isolated Spoken Word Recognition）**とは，単語が 1 つだけ含まれていて，単語の前後に無音区間がある音声を認識する方式のことです．

この章の実習を実現するための信号処理とパターン認識のコマンドは C 言語で作られています．On-The-Fly 音声認識のコマンド（recog）は，シェルスクリプトです．PulseAudio の録音コマンド（rec），音声区間検出（vad コマンド，第 6.3.2 節），スペクトル分析（mfcc コマンド，第 3 章，第 1 日の実習で作成），および、Viterbi アルゴリズム（第 5.3.3 節）による一方向性連続 HMM（第 5.4 節）に対する確率計算（_recog コマンド）等を組み合わせて実現しています．On-The-Fly 音声認識に関わっているソースコードを以下に列挙します．実習作業に余裕があれば，内容を見て処理の仕組みの理解に努めてください．

wrecog/program/recog On-The-Fly 単語認識．以下の 5 つのコマンドを組み合わせているシェルスクリプト．

sound/audioIN 音声入力（録音）コマンド．シェルスクリプト．

sound/vad.c 音声区間検出．C 言語．

wrecog/program/mfcc.c スペクトル（MFCC）分析．C 言語．

wrecog/program/_recog.c Viterbi アルゴリズム．C 言語．

sound/hupAudioIN 音声入力プロセスを終了する．シェルスクリプト．

6.5.8 レポート（第 3 週）

1. 単語認識実験
   (a) 単語認識タスクを設計してください．
      - あなたの単語認識タスクの目的を述べてください．仮に使うとしたらどのような用途に用いますか？
      - 5 種類以上の単語からなる単語集合を定義し，一覧表（表 6.1 の形式）を作ること．
      - 各単語のモデルに用いた HMM の状態数を上記一覧表に記述すること．
   (b) HMM の学習曲線（図 6.5 に相当するもの）．認識対象の単語のうち任意の 1 つに対するものでよい．
   (c) 学習の検証の結果
   (d) On-The-Fly 単語認識の結果と考察
2. このテーマについて（第 1 週から第 3 週の総括）
   (a) ポイントは何であったか？
   (b) 良くわかったこと
   (c) わからなかったこと
   (d) 要望
   (e) 感想，その他