卒業論文

隠れマルコフモデルによる時間領域での音声信号の学習と生成

76050200　黒田 優士

指導教員　伊達 章 准教授

2009 年 2 月

宮崎大学 工学部 情報システム工学科

概要

1995 年，Künsch らにより “Hidden markov random fields” と題する画期的な論文が発表された [1]．この論文では，単純な隠れマルコフモデルが，学習データに関する任意の確率分布をいくらでも精度良く表現する能力をもつことが証明されている．隠れマルコフモデルは確率的生成モデルである．例えば，学習データとして時間領域の音声波形を考える．学習データの確率分布を適切にモデル化することができれば，音声認識と音声合成が同一のモデルで実現できる．本研究では，第一に，Künsch らの実験のアイデアをもとに，時間領域の音声波形を学習させた．次に，学習と同一のモデルを用いて音声を生成することを試みた．隠れマルコフモデルは数学的には任意の確率分布を近似的に表現可能なモデルであっても，実際の表現能力については不明な点が多い．この点を重点的に調べた．その結果，音声信号の特徴を捕らえた学習に成功した．

目次

第 1 章

はじめに

1.1 研究の背景

通常，音声研究では時間領域の音声波形はいったん短時間フーリエ変換され，周波数領域で特徴量が分析される．一度，周波数領域で表現された音声信号[*1]を時間領域に戻すことは難しい．なぜなら，周波数領域では音声の情報が失われているからである．そのため，音声信号を生成するためには別のモデルを考える必要がある．

1995 年，Künsch らにより “Hidden markov random fields” と題する画期的な論文が発表された [1]．いま，$X = X_1, X_2, \cdots$ をマルコフ性をもつ確率過程とする．ある関数 f を使い $Y_t = f(X_t)$ として得られる $Y = Y_1, Y_2, \cdots$ を隠れマルコフ過程という．論文 [1] では，図 1.1 のような単純な隠れマルコフモデルが，Y に関する任意の確率分布をいくらでも精度良く表現する能力をもつことが証明されている．隠れマルコフモデルは確率的生成モデルである．例えば，Y として音声信号を考える．Y の確率分布を適切にモデル化することができれば，音声認識と音声合成が同一のモデルで実現できる．また，隠れマルコフモデルは音声の特徴を確率で表現するので，同じモデルで何度生成しても毎回違う波形ができる．人間が「こんにちわ」と 10 回言ってみる．10 回とも「こんにちわ」と聞こえるだろう．けれど，その音声信号は毎回違う波形をしている．このことから，隠れマルコフモデルは人間に近い生成モデルであると考えられる．

1.2 目的

本研究では，第一に，Künsch らの実験のアイデアをもとに，音声信号を学習させた．次に，学習と同一のモデルを用いて音声信号を生成することを試みた．隠れマルコフモデルは数学的には任意の確率分布を近似的に表現可能なモデルであっても，実際の表現能力については不明な点が多い．この点を重点的に調べた．その結果，音声信号の特徴を捕らえた学習に成功した．

[*1] 本論文では，時間領域での音声波形を音声信号と呼ぶ．

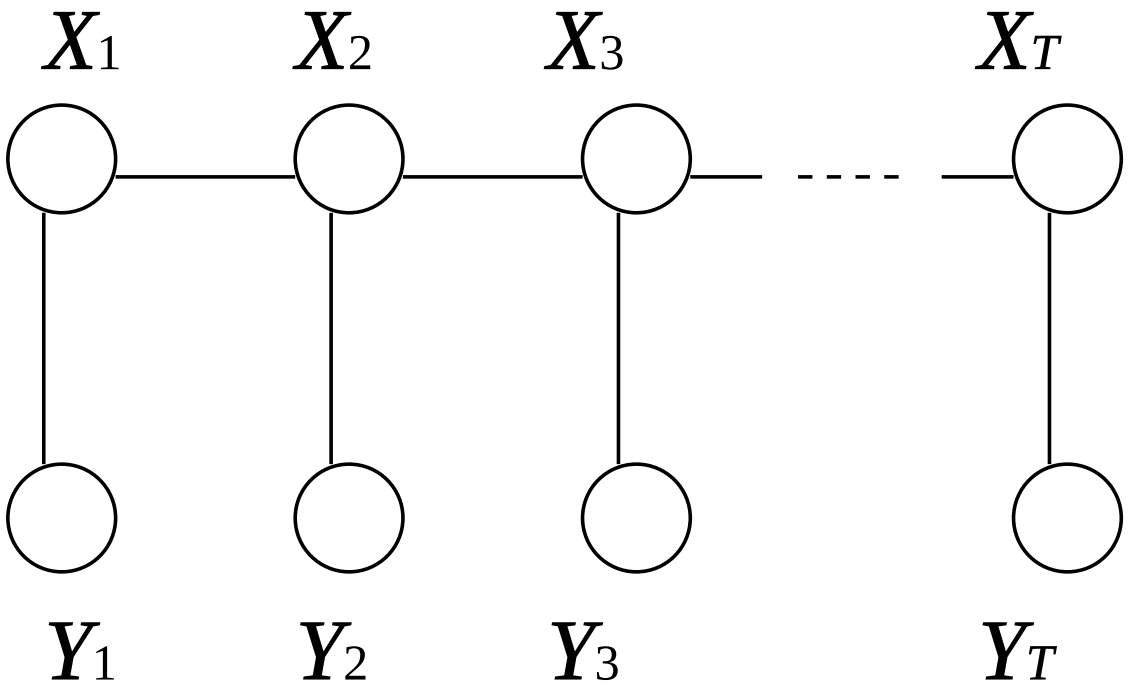

図 1.1. 隠れマルコフモデル．確率変数の依存性を示すグラフ $p(X, Y)$.

1.3　本論文の構成

まず，2 章で確率的生成モデルである隠れマルコフモデルについて紹介する．また，隠れマルコフモデルの計算アルゴリズムの EM アルゴリズムと Backward-Forward アルゴリズムについても述べる．3 章ではモデルの学習能力および生成能力を音声信号を用い，調べた結果を示す．4 章では全体のまとめを書く．

第 2 章

モデルとアルゴリズム

2.1 モデル

2.1.1 マルコフチェーン

隠れマルコフモデルはマルコフチェーンをその基礎としている．まず，そのマルコフチェーンについて述べる．

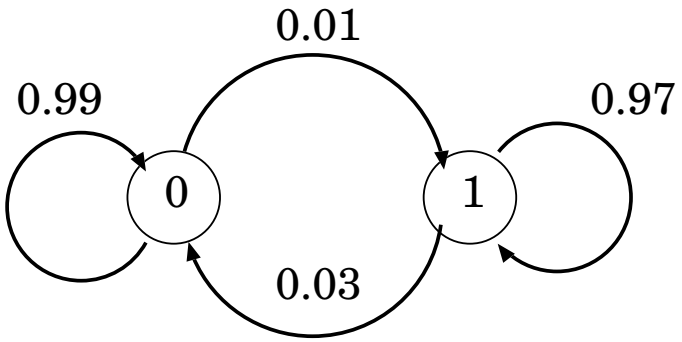

図 2.1. 状態遷移図

図 2.1 に示すようなマルコフ的情報源を考える．この情報源は 0, 1 の 2 つの状態をもち，状態間を時々刻々遷移する．状態は丸で囲われており，状態遷移の確率は状態間に引かれた矢印のついた線の上に書かれている．図では状態 0 にあったときに，次もまた状態 0 のままでいる確率が 0.99，状態 1 に移る確率が 0.01 であることを示している．このような確率的な状態遷移を 500 回繰り返すと，出現する可能性のある系列は 2^{500} 通りある．その中には $0000\ldots000$ というものから，$010101\ldots0101$ という滅多なことでは見ることができない系列も含まれる．

初期状態として，時間 $t = 0$ において，状態 0 にいる確率を $p_0 = 0.5$ とする．確率は足し算すると 1 であるから，$p_1 = 0.5$ である．時間 t で $X_t = i$ という状態にいた場合に，時間 $t+1$ で $X_{t+1} = j$ という状態に遷移する確率 $\mathrm{Prob}\{X_{t+1} = j | X_t = i\}$ を p_{ij} と書く．この p_{ij} を

並べてつくった $P = \{p_{ij}\}$ という行列を遷移行列という．図 2.1 の場合，

$$\begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{11} \end{bmatrix} = \begin{bmatrix} 0.99 & 0.01 \\ 0.03 & 0.97 \end{bmatrix}$$

と書ける．

T 回遷移を繰り返すとして，このマルコフチェーンから生成される最も尤もらしい 0 と 1 の系列は，0000...000 とすべて 0 である系列が最も高い確率 $0.5 \times (0.99)^T$ で出現する．現在の状態に依存して，確率的に状態が遷移していく過程をマルコフ過程という．図 2.2 はマルコフ過程の確率変数の依存性を示している．各ノードが確率変数を表し，線でつながっているのは，依存性があることを示している．例えば，X_1 から X_T までの関係を調べると，X_3 のとりうる値は X_2 と X_4 にのみ依存していることがわかりやすい．つまり，

$$p(X_3|X_1, X_2, X_4, \ldots, X_T) = p(X_3|X_2, X_4)$$

となる．

2.1.2　隠れマルコフモデル

図 1.1 の各ノードが確率変数を表し，線でつながっているのは，依存性があることを示している．$X = X_1, X_2, \ldots, X_t, \ldots, X_T$ を内部データと呼ぶ．観測できるのは X ではない．$Y_t = X_t \bmod 5$ のような，X から Y は一つに決まるが，Y から X は一つに決められない関数を用いて得られる $Y = Y_1, Y_2, \ldots, Y_T$ である．一般的に Y はマルコフ過程ではない．マルコフ性をもたない Y が観測でき，マルコフ性をもつ X が隠されている．このようなモデルを隠れマルコフモデルという．

図 1.1 に示すモデルを使う利点は 2 つある．一つは，観測値 Y が与えられた後は，X の確率分布 $p(X_1, X_2, \ldots, X_T|Y)$ の構造は単なる線型のグラフ (図 2.2) で表現できることである．単純な構造があるので，様々な量，例えば $p(X_2, X_3|Y)$ などが楽に計算できる．もう一つは，観測値 Y の周辺確率分布 $p(Y)$ が完全結合型のグラフ（図 2.3）で表現できることにある．図 2.3 のような完全結合型の依存性を持つモデルであれば，確率変数間 $Y_1, Y_2, \ldots, Y_T$ にどんな依存性がある同時確率分布 $p(Y_1, Y_2, \ldots, Y_T)$ も，いくらでも精度良く近似しモデル化できる．Y は観測データなので推定する必要はない．一方，内部データ X は観測値 Y に対する解釈を表現する．そのため，X の周辺分布をはじめ，様々な X の関数を推定したい．図 1.1 の隠れマルコフモデルは，この目的を満足させてくれる．

隠れマルコフモデルの目的は Y を観測し，もとの X を推定することである．つまり，Y を観測し，最も尤もらしい $\hat{X} = \hat{X_1}, \hat{X_2}, \ldots, \hat{X_T}$ を求めることである．これを式で書くと，

$$\hat{X} = \arg\max_X \mathrm{Prob}(X|Y) \tag{2.1}$$

となる．$\arg\max$ という記号は，条件付き確率 $\mathrm{Prob}(X|Y)$ を最大にする X の組み合わせという意味である．

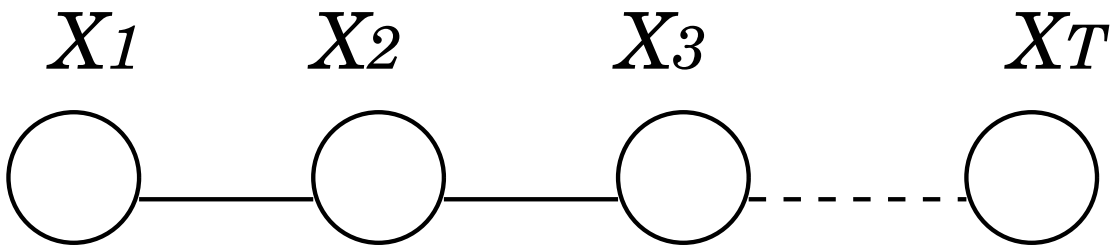

図 2.2. マルコフ過程．確率変数の依存性：$p(X|Y)$

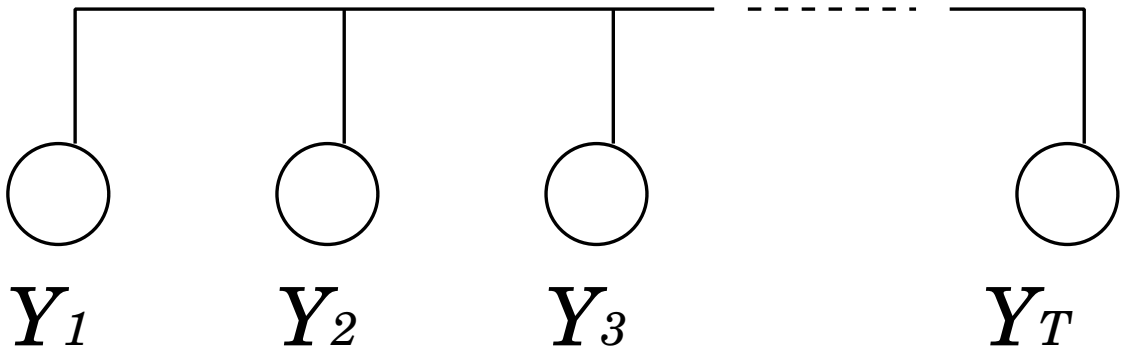

図 2.3. 確率変数の依存性：$\sum_X p(X,Y) = p(Y)$

2.2 学習アルゴリズム

最も尤もらしい $\hat{X}$ を求めるためのアルゴリズムを説明する．内部データ X が $1,\ldots,N$ の値をとるとする．時間 t で値 $X_t = i$ という状態にいた場合に，時間 $t+1$ で $X_{t+1} = j$ という状態へ遷移する確率 $\mathrm{Prob}\{X_{t+1} = j | X_t = i\}$ を p_{ij} と書く．この p_{ij}, $i,j \in \{1,\ldots,N\}$ をパラメータと呼ぶことにする．$\hat{X}$ を求めるには，$\hat{X}$ を得るための最適なパラメータを求める必要がある．一般的に，最適なパラメータの値はわかっていない．この場合，パラメータの値を観測値から推定する．最適なパラメータを一回で求めるのは難しい，よって，パラメータをランダムに設定し，収束するまで繰り返し計算する．そのための式は次のようになる．

$$p_{ij} = \frac{\sum_{t=1}^{T} p(X_t = i, X_{t+1} = j, Y)}{\sum_{a=1}^{N}\sum_{t=1}^{T} p(X_t = i, X_{t+1} = a, Y)} = \frac{\sum_{t=1}^{T} p(X_t = i, X_{t+1} = j, Y)}{\sum_{t=1}^{T} p(X_t = i, Y)} \tag{2.2}$$

この式を効率良く計算したい．本研究では，この p_{ij} は時間的には変動しないことにする．具体的に説明するため，$T = 5$ とする．$\mathbf{y}$ は与えられている．遷移する回数は $4(= T-1)$ である．まず，式 (2.2) の分母を考える．

$$\begin{aligned}\sum_{t=1}^{4} p(X_t = i, \mathbf{y}) &= p(X_1 = i, \mathbf{y}) + p(X_2 = i, \mathbf{y}) + p(X_3 = i, \mathbf{y}) + p(X_4 = i, \mathbf{y}) \\ &= \sum_{\tilde{x_2},\tilde{x_3},\tilde{x_4},\tilde{x_5}} p(i, \tilde{x_2}, \tilde{x_3}, \tilde{x_4}, \tilde{x_5}, \mathbf{y}) + \sum_{\tilde{x_1},\tilde{x_3},\tilde{x_4},\tilde{x_5}} p(\tilde{x_1}, i, \tilde{x_3}, \tilde{x_4}, \tilde{x_5}, \mathbf{y})\end{aligned}$$

$$+\sum_{\tilde{x_1},\tilde{x_2},\tilde{x_4},\tilde{x_5}} p(\tilde{x_1},\tilde{x_2},i,\tilde{x_4},\tilde{x_5},\mathbf{y}) + \sum_{\tilde{x_1},\tilde{x_2},\tilde{x_3},\tilde{x_5}} p(\tilde{x_1},\tilde{x_2},\tilde{x_3},i,\tilde{x_5},\mathbf{y})$$

$\tilde{x_t}$ は値が定まっていない変数である．4 つの項がでてきた．$X_3 = i$ の場合を計算する方法を示す．

$$p(x_3 = i, \mathbf{y}) = \sum_{\tilde{x_1},\tilde{x_2},\tilde{x_4},\tilde{x_5}} p(\tilde{x_1},\tilde{x_2},i,\tilde{x_4},\tilde{x_5},\mathbf{y})$$

各 X_t が $|N|$ 通りの値をとりうる場合，$|N|^{T-1} = |N|^4$ 個の項を足す必要がある．$T = 5$ ではなく $T = 500$ などになると，この計算は難しい．この場合，変数間の依存性を示すグラフを用いると計算がしやすい．図 2.4 は $X_3 = i$ のときの依存性を示している．黒いノードは値が固定されている．$\mathbf{x}, \mathbf{y}$ の同時分布 $p(\mathbf{x}, \mathbf{y})$ は

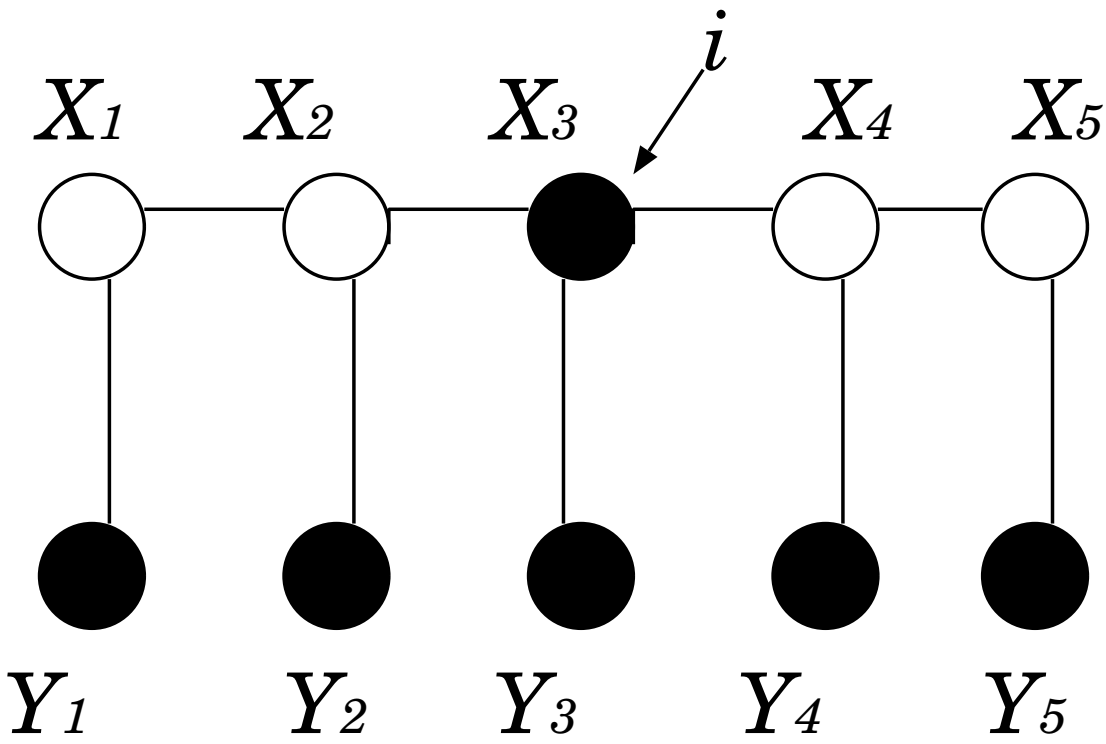

図 2.4. $X_3 = i$ のときの変数間の依存性を示すグラフ：黒いノードは値が固定されている

$$p(\mathbf{x}, \mathbf{y}) = \mathrm{Prob}(x_1, \ldots, x_T, y_1, \ldots, y_T) = p_{x_1} p_{x_1 x_2} \cdots p_{x_{T-1} x_T} q_{x_1 y_1} q_{x_2 y_2} \cdots q_{x_T y_T}$$

と掛け算の式で記述できる．そこで，$L_3(i), R_3(i)$ という量を次のように定義する．

$$p(x_3 = i, \mathbf{y}) = \sum_{\tilde{x_1},\tilde{x_2},\tilde{x_4},\tilde{x_5}} p(\tilde{x_1},\tilde{x_2},i,\tilde{x_4},\tilde{x_5},\mathbf{y}) = L_3(i) R_3(i)$$

具体的には，$L_3(i)$ は X_1，X_2 の値に関係なく，$X_3 = i$ となるような 6 本のリンクからなる条件付き確率を掛け算した値を，$|N|^2$ 通り足し合わせた数である．同様に，$R_3(i)$ は X_4，X_5

の値に関係なく，$X_3 = i$ となるような条件付き確率を掛け算した値を, $|N|^2$ 通り足し合わせた数である．L_{t+1} と L_t の関係性などは

$$L_{t+1}(i) = \sum_j L_t(j) p_{ji}^m g(i, y_{t+1}) \tag{2.3}$$

$$R_t(i) = \sum_j R_{t+1}(j) p_{ij}^m g(j, y_{t+1}) \tag{2.4}$$

と書ける．L_1, R_T はそれぞれ

$$L_1(i) = p_i^m g(i, y_1) \tag{2.5}$$

$$R_t(i) = 1 \tag{2.6}$$

とする．ここで, $g(i, y_3)$ は, 例えば $i \in \{1, 2, \ldots, 30\}, y_3 \in \{1, 2, \ldots, 8\}$ の場合, $1 + i \bmod 8 = y_3$ となっていれば 1，なっていなければ 0 をとる関数である．p_i^m は $X_1 = i$ となる確率である．図 2.5，図 2.6 より，$L_1(i), L_2(i), \ldots$ を順に, および $R_T(i), R_{T-1}(i), \ldots$ を順に計算していけばよいことがわかる． 分母はこのように計算できる．式 (2.2) の分子についても，分母と

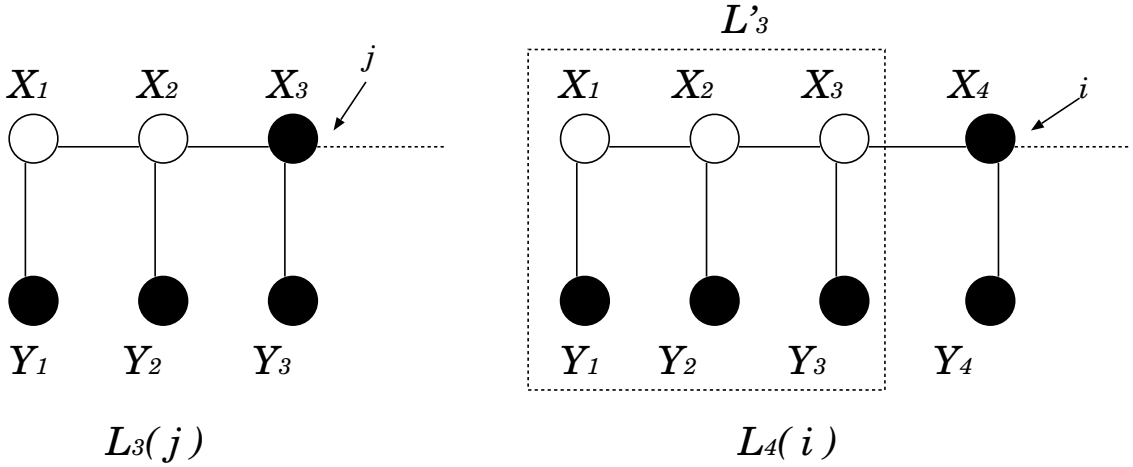

図 2.5. $L_4(i) = \sum_j L_3(j) p_{ji}^m g(i, y_4)$．黒いノードは値が固定.

同様に考えることができる．図 2.7 より,

$$p(X_t = i, X_{t+1} = j, \mathbf{y}) = L_t(i) p_{ij} g(j, y_{t+1}) R_{t+1}(j) \tag{2.7}$$

と書ける．$t = 1$ および $t = T - 1$ の場合も表現できている．よって，式 (2.2) は

$$p_{ij}^{m+1} = \frac{\sum_{t=1}^{T-1} L_t(i) p_{ij}^m g(j, y_{t+1}) R_{t+1}(j)}{\sum_{t=1}^{T-1} L_t(i) R_t(i)} \tag{2.8}$$

と書けるので，これを計算すればいい．p_{ij}^m を使い p_{ij}^{m+1} が計算できた.

また，初期状態として X_1 がとりうる値について，それぞれの確率を計算しておく必要がある．特定の学習データに依存しない推定値を求めたい．そこで,

$$p_i^{m+1} = \frac{\sum_{t=1}^{T} L_t(i) Rt(i)}{\sum_a \sum_{t=1}^{T} L_t(a) R_t(a)}$$

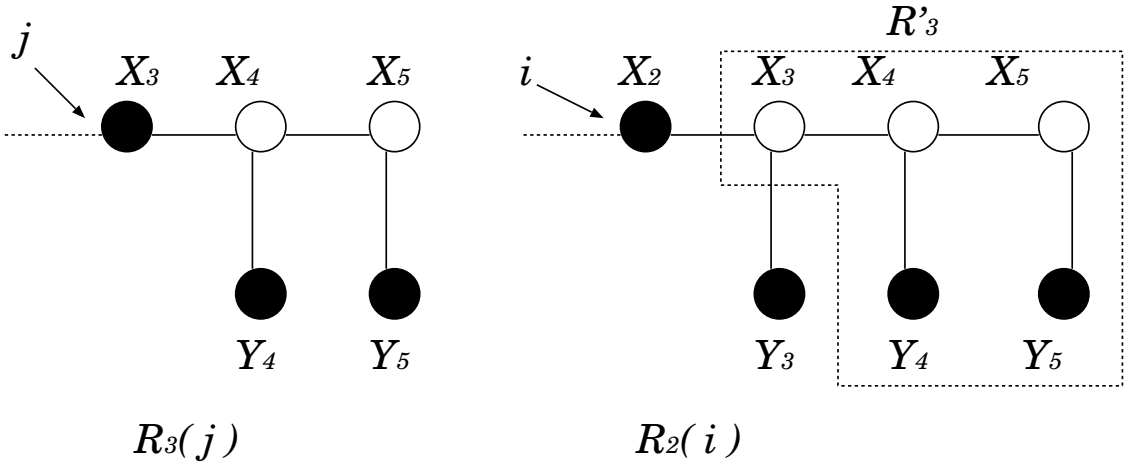

図 2.6. $R_2(i) = \sum_j R_3(j) p_{ij}^m g(j, y_3)$. 黒いノードは値が固定.

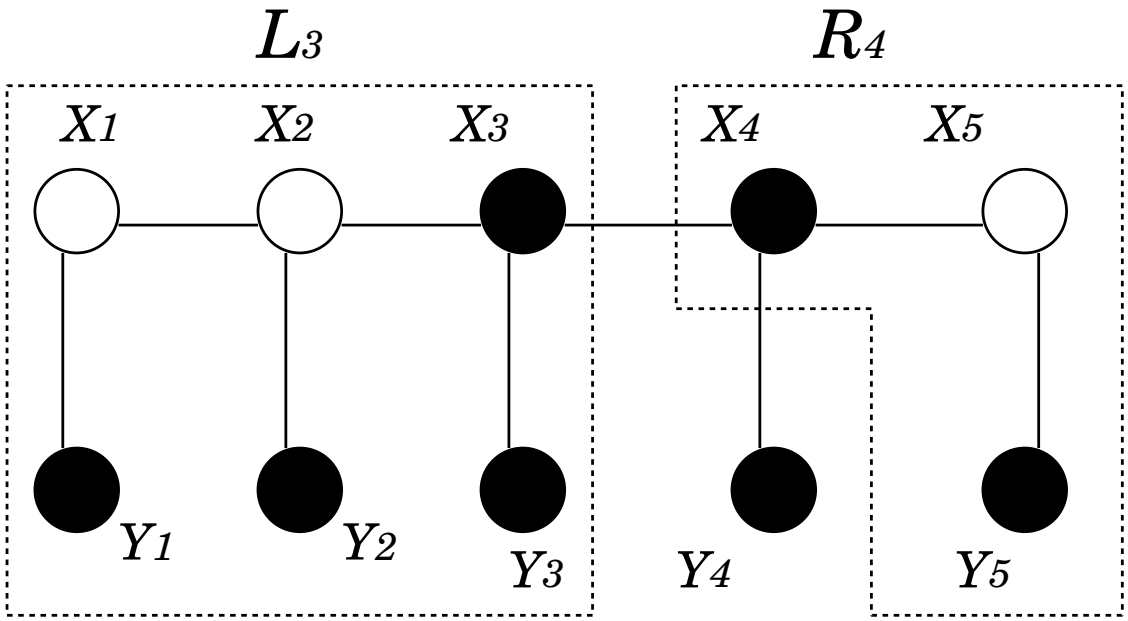

図 2.7. $p(x_3 = i, x_4 = j, \mathbf{y}) = L_3(i) p_{ij} g(j, y_4) R_4(j)$

と，状態 i にいる確率を計算する.

以上のことから，あらかじめ式 (2.3) と式 (2.4) を求めておく必要があることがわかる．しかし，通常，計算するとアンダーフローが起こり，コンピュータで計算できない．それを回避する方法を以下に示す.

1. まず，$L_1(i) = p_i^0 g(i, y_1), i = 1, 2, \ldots$ を計算する.
2. 次に $l_1 = \sum_i L_1(i)$ を計算する.
3. $L_1(i) := L_1(i)/l_1, i = 1, 2, \ldots$ と足して 1 になるように変換する.
4. $L_1(j), j = 1, 2, \ldots$ を使い $L_2(i)$ を計算し，l_2 を求め，同様に (足して 1 になるように)

変換する．

5. $L_T(i), i = 1, 2, \ldots$ まで計算して，$l_1, \ldots, l_T$ を求める．
6. $R_T(i) = 1$ からはじめて，L_t と同様に $R_1(i)$ まで計算する．
   ここで
   $$R_t(i) := R_t(i)/l_t$$
   と変換する．
7. 式 (2.8) により p_{ij}^{m+1} を求める．ただし，
   $$p_{ij}^{m+1} = \frac{p_{ij}^{m+1}}{\sum_k p_{ik}^{m+1}}$$
   として，確率が足して 1 になるように正規化する．

以上の計算は，動的計画法を用いた計算方法である．これらの式を使い実験をする．

第 3 章

実験

論文 [1] のなかで，Künsch らが提示している実験をする．論文が書かれたのは 1995 年である．当時は現在と比べてコンピュータの性能が低かった．そのため，学習データ数が多いと計算できなかった．そこで, 本論文ではデータ数の多い学習データを使い, モデルの学習能力と生成能力を調べた．実験は，母音と子音を学習させ，それぞれのパラメータから音声信号を生成した．

3.1 母音の学習

この節では，母音をサンプリング周波数を 2kHz，10kHz と変えてパラメータを推定した．また，内部状態数が多いと，複雑な確率分布でも表現できることを確かめた．そして，データ数と内部状態数による生成能力の違いを調べた．

まず，「あ」[*1]の音声を量子化ビット数 3bit（8 値），2kHz でサンプリングした音声信号を学習データとして与えた（図 3.1）．学習データ数は 200 個である．$Y_t = X_t \bmod 8$, $t = 1, 2, \ldots, 200$ を使い, パラメータ p_{ij} の初期値は $\sum_j p_{ij} = 1$, $0 \leq p_{ij} \leq 1$, $i, j \in \{1, 2, \ldots, N\}$ となるように乱数で設定した．N を $10, 20, 30, 40, 50, 60$ と変えて実験した．

学習の結果求められたパラメータを使い，区間 $[0.0,\ 1.0)$ の一様乱数を用いて X を生成した．X から得られた Y のグラフを図 3.2 に示す．図 3.1 と図 3.2 のそれぞれの波形を比較する．図 3.2 では，N が大きくなるにつれて図 3.1 に近くなっているようにみえる．$N = 60$ で学習データの特徴が少しでている．

Künsch らは $N = 60$ で実験をやめている．そして, データ数 200 個に対し, $60 \times 60 = 3600$ 個のパラメータを推定していることから，過学習が起きていると考えた．しかし，生成された波形が学習データと似ていないことから，本当に過学習が起きているのか疑わしい．そこで，さらに N を大きくしてパラメータを推定した．$N = 80$, 90, 100, 200, 300, 400 で推定する，その結果を図 3.3 に示す．どれも $N = 60$ のときよりも，図 3.1 に似た波形になっているように思える．特に，$N = 100$ 以上が学習データの波形によく似ている．つまり，過学習

[*1] 本論文では，学習に使われる音声信号を「あ」，“k” のように表記し，生成された音声信号を [a]，[k] のように発音記号で表記する．

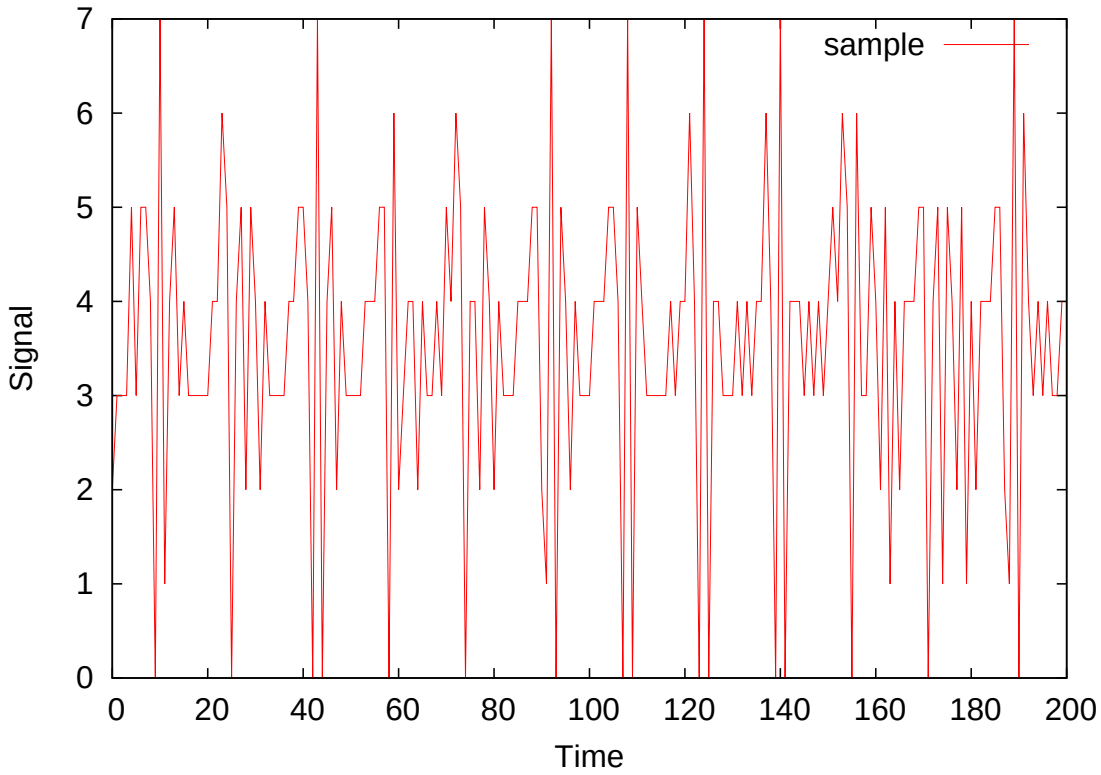

図 3.1. 2kHz の学習データ [a] の波形．横軸は t，縦軸は Y_t がとる値を示している．

は $N = 100$ で起きているのではないかと考えられる．また，この実験では 200 個の学習データで，最大 16 万個のパラメータを推定している．これは明らかに無謀である．よって，学習データの数を多くしてパラメータを推定した．

次に，10kHz でサンプリングした音声信号を学習データとした．データ数は 2250 個である．学習データの $t = 0, \ldots, 1000$ の部分を図 3.4 に示す．内部データ X がとる値の範囲 N を 10, 30, 60, 80, 100, 200 としてパラメータを推定した．

それぞれの推定されたパラメータから得られる，Y のなかの一つを図 3.5 に示す． 学習データの波形に似た波形はない．そこで，モデルが特徴を正確に学習しているのか確かめるために，「い」「う」「え」「お」でも同様に実験した．それぞれ，$N = 60$ と $N = 100$ で学習したパラメータから生成した波形の一つを図 3.7 から図 3.13 に示す．それぞれの図の上が学習データ，左が $N = 60$，右が $N = 100$ のときの波形である．図をみると，それぞれの学習データの波形に似た波形はないが，まったく違う波形ができているわけではないのがわかる．例えば，[i] は明らかに他の学習データとは違う波形ができている．生成された波形が学習データごとに判別できることから，それぞれの学習データで学習したモデルは特徴をつかむことができていると考えられる．

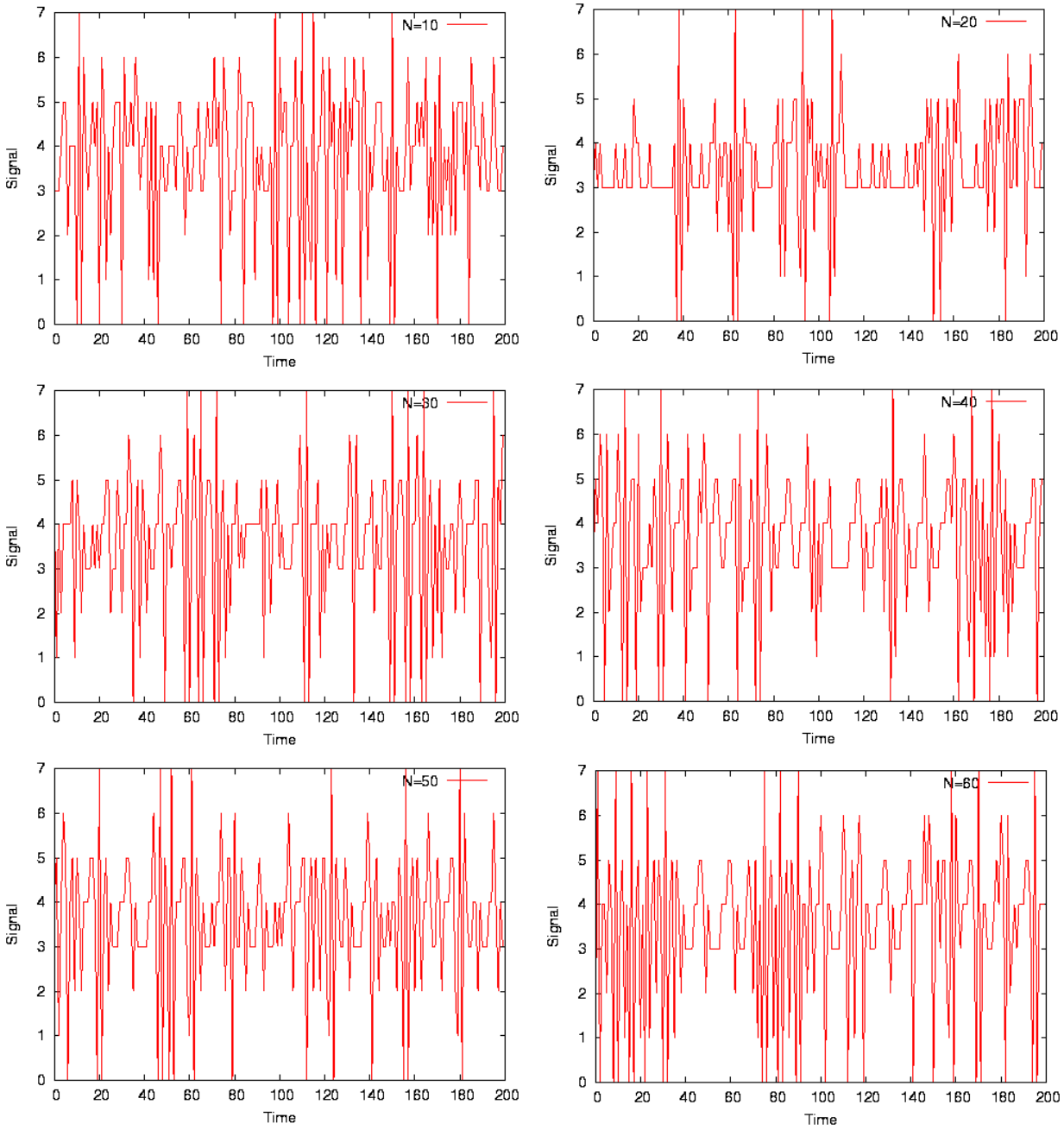

図 3.2. 生成された [a] 波形（2kHz）のなかの一つ．左から右の順に，一番上が $N = 10$, 20, 真ん中が $N = 30$, 40, 一番下が $N = 50$, 60 である．横軸が時間 t, 縦軸は Y_t の値を表している．

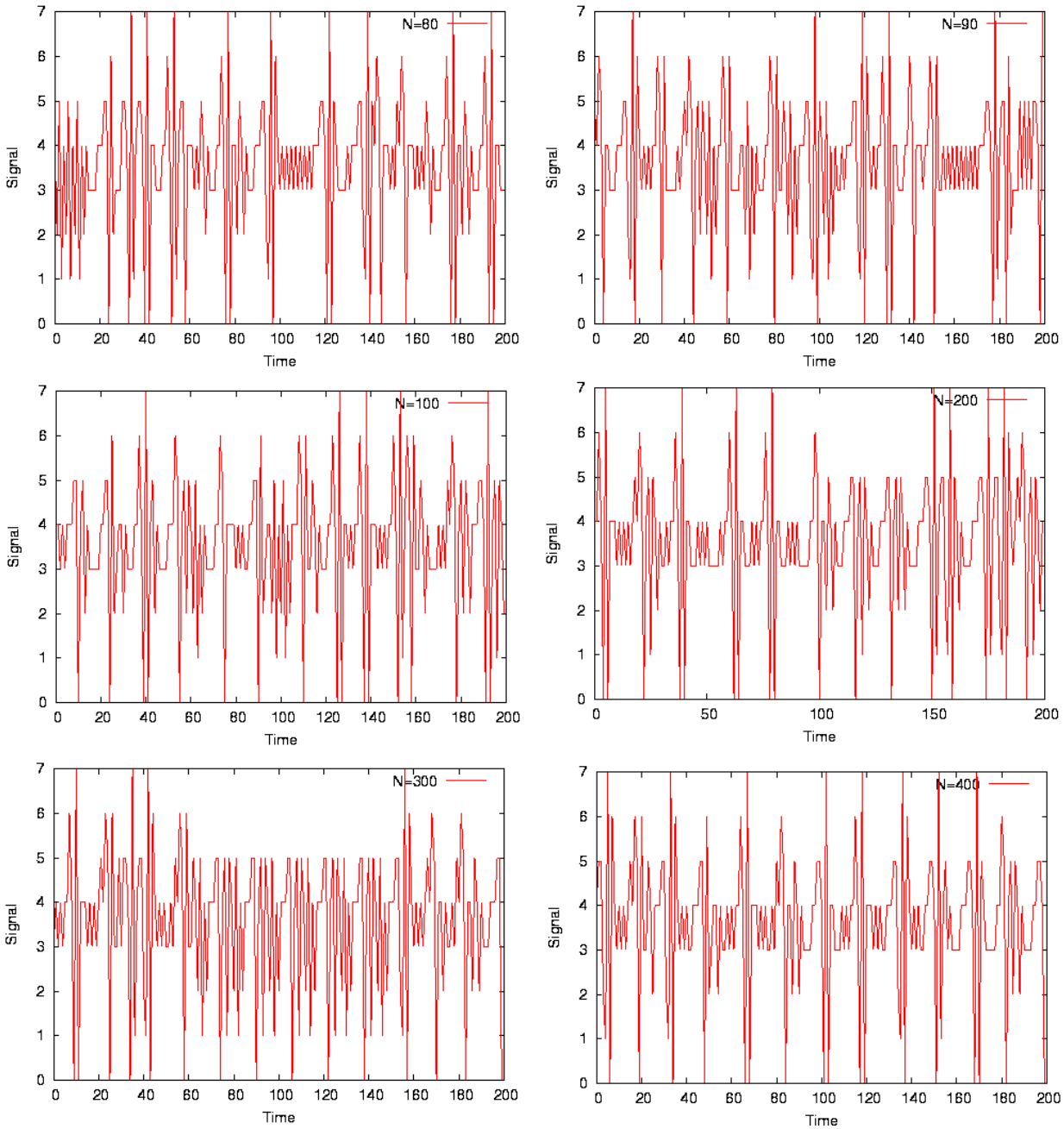

図 3.3. 生成された [a] 波形（2kHz）のなかの一つ．左から右の順に，上：$N = 80$, 90，中：$N = 100$, 200，下：$N = 300$, 400 である．横軸が時間 t，縦軸は Y_t の値を表している．

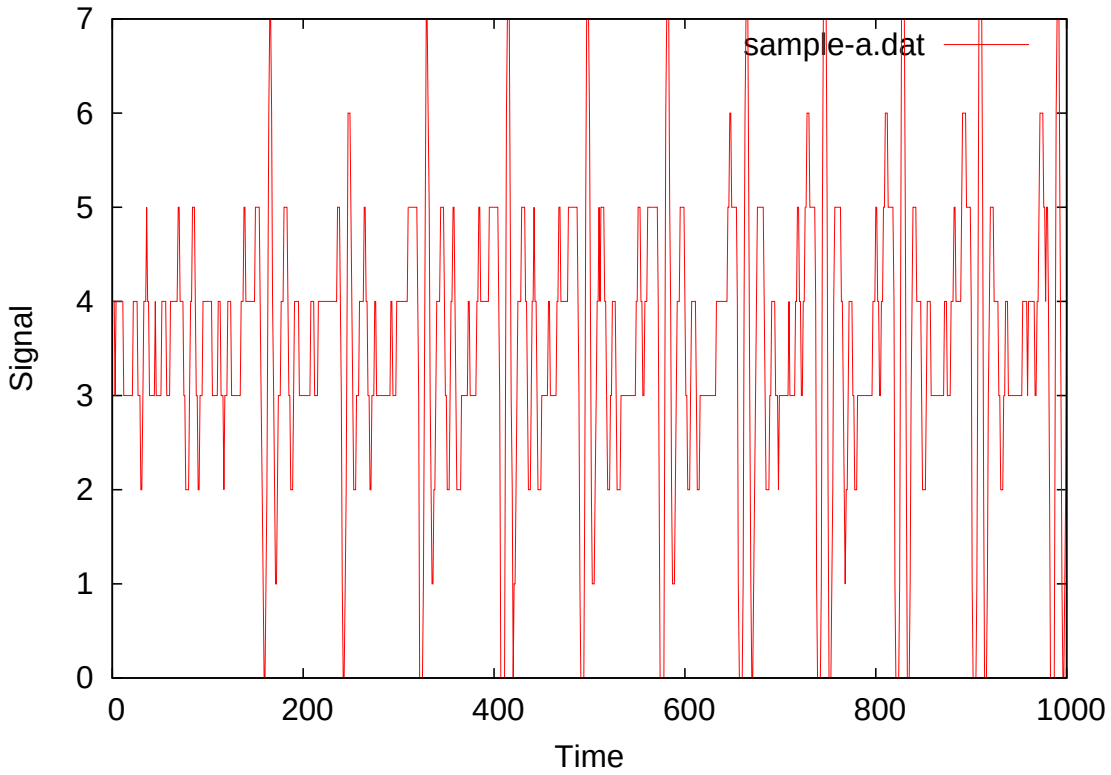

図 3.4. 10kHz の学習データ [a] の波形の $t = 0, ..., 1000$ の部分．横軸は時間 t，縦軸は Y_t の値をあらわしている．

3.2　子音の学習

ここでは，X の状態数 N が多いと複雑な確率でも表現できることを確かめる．子音の図 3.14 は子音 “k”，“m” の学習データの一部を示している．“k” の学習データは，「か」「き」「く」「け」こ」を量子化ビット数 3（8 値），10kHz でサンプリングし，それぞれの音声信号から子音と思われる部分を抜き取り，一つのデータとした．“m” についても同様である．データ数は，“k” が 3786 個，“m” が 2911 個である．

$N = 10, 30, 40, 50, 60, 100$ での学習の結果得られたパラメータにより，生成できる［ k ］，[m] の波形のなかの一つを図 3.15，図 3.16 に示す． 子音の場合も，母音の学習と同じく，[k] と [m] の違いがわかる波形ができている．それぞれの特徴をとらえた学習と生成ができている．このことより，隠れマルコフモデルは学習データが複雑な構造をもつ場合でもパラメータが計算できることが確認できた．

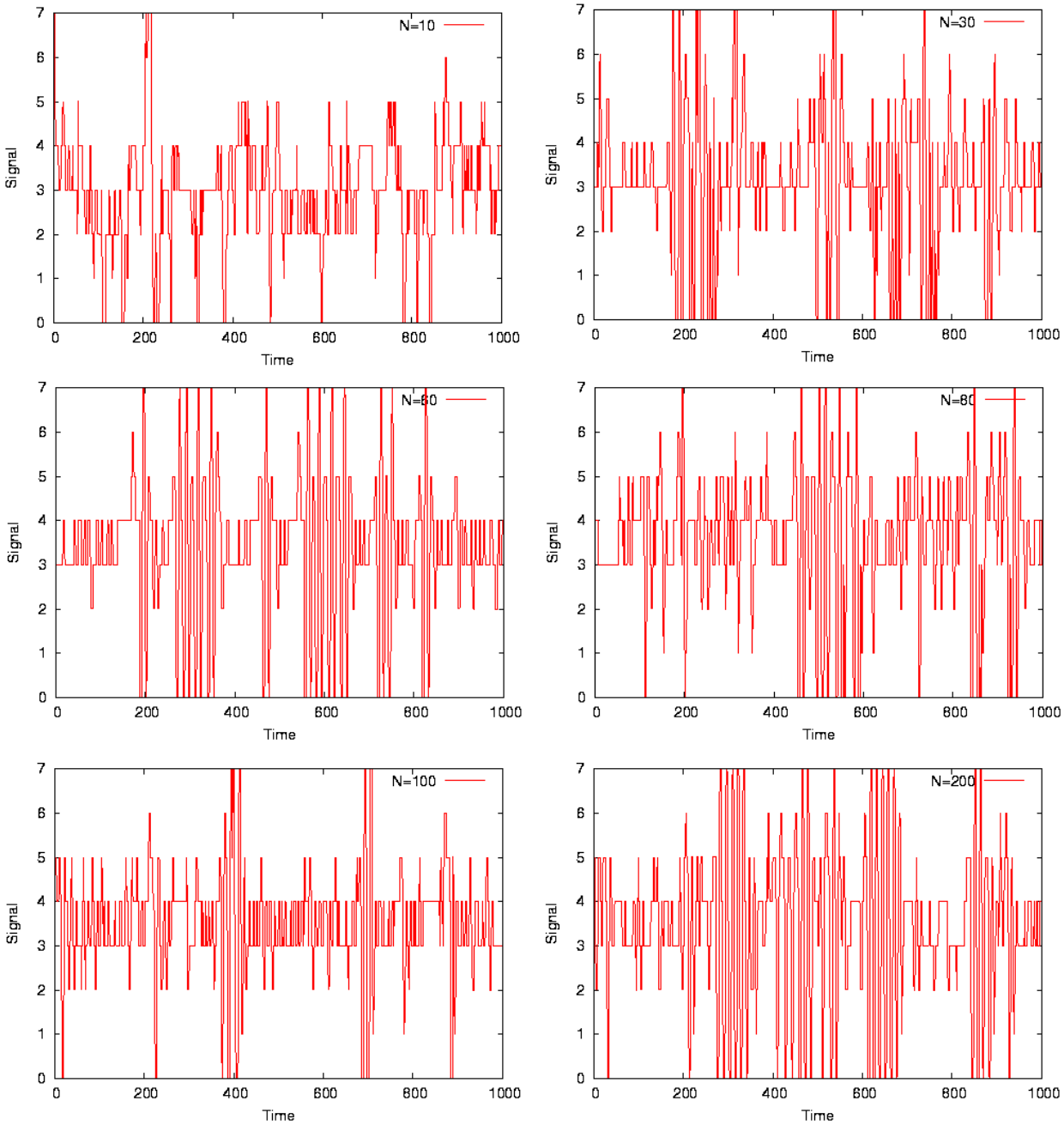

図 3.5. 生成された [a] の波形 (10kHz) の一つ．左から右の順に，上：$N = 10$, 30, 中：$N = 60$, 80, 下：$N = 100$, 200 である．横軸は時間 t, 縦軸は Y_t の値を示している．

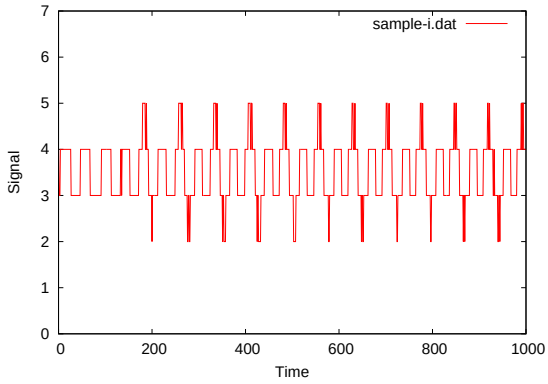

図 3.6. 学習データ [i] の波形（10kHz）の t=0,...,1000 の部分

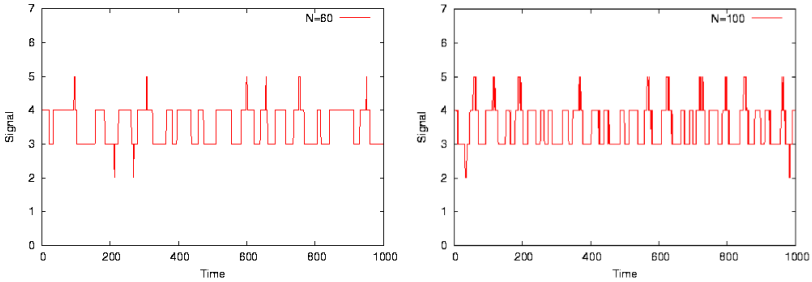

図 3.7. 生成された [i] の波形（10kHz）の一つ．左：$N = 60$, 右：$N = 100$ である．横軸は時間 t，縦軸は Y_t の値を示している．

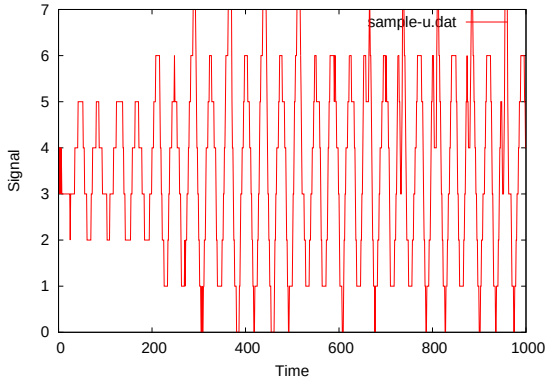

図 3.8. 学習データ [u] の波形（10kHz）の t=0,...,1000 の部分

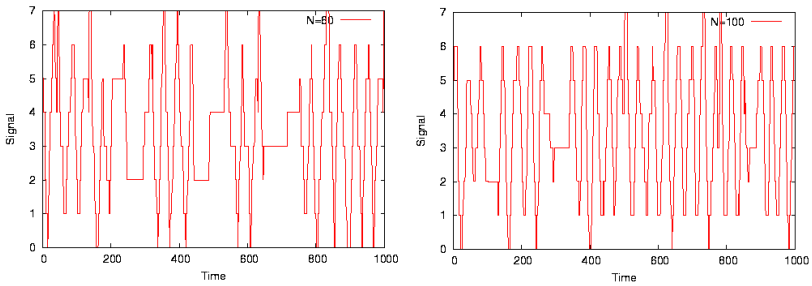

図 3.9. 生成された [u] の波形（10kHz）の一つ．左：$N = 60$, 右：$N = 100$ である．横軸は時間 t，縦軸は Y_t の値を示している．

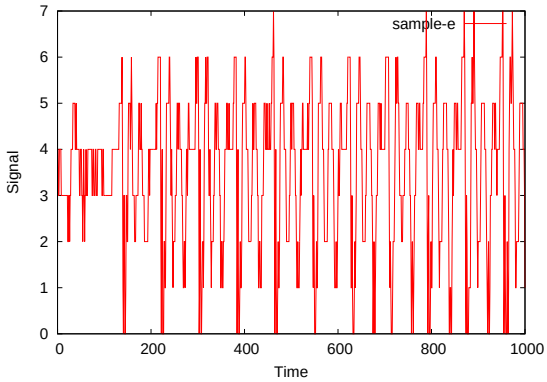

図 3.10. 学習データ [e] の波形（10kHz）の t=0,...,1000 の部分

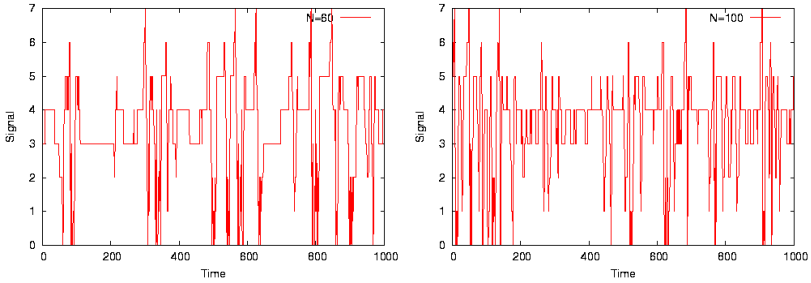

図 3.11. 生成された [e] の波形（10kHz）の一つ．左：$N = 60$, 右：$N = 100$ である．横軸は時間 t，縦軸は Y_t の値を示している．

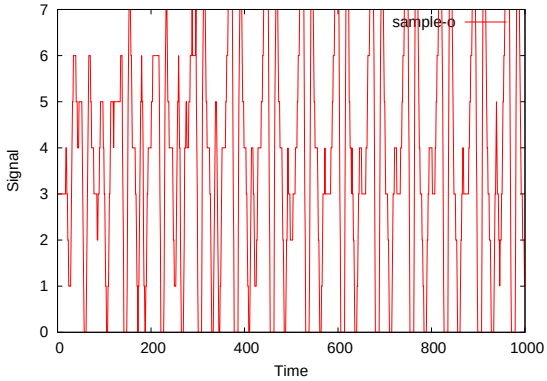

図 3.12. 学習データ [o] の波形（10kHz）の t=0,...,1000 の部分

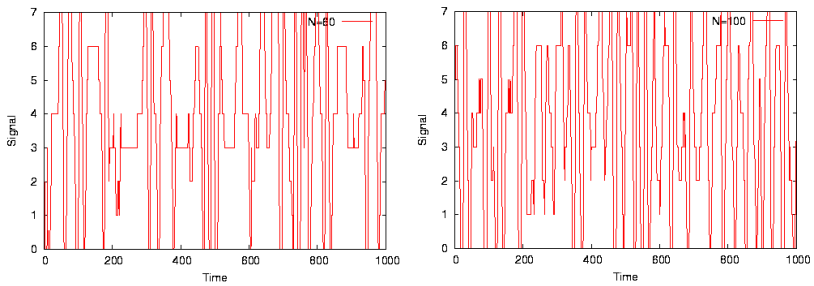

図 3.13. 生成された [o] の波形（10kHz）の一つ．左：$N = 60$, 右：$N = 100$ である．横軸は時間 t，縦軸は Y_t の値を示している．

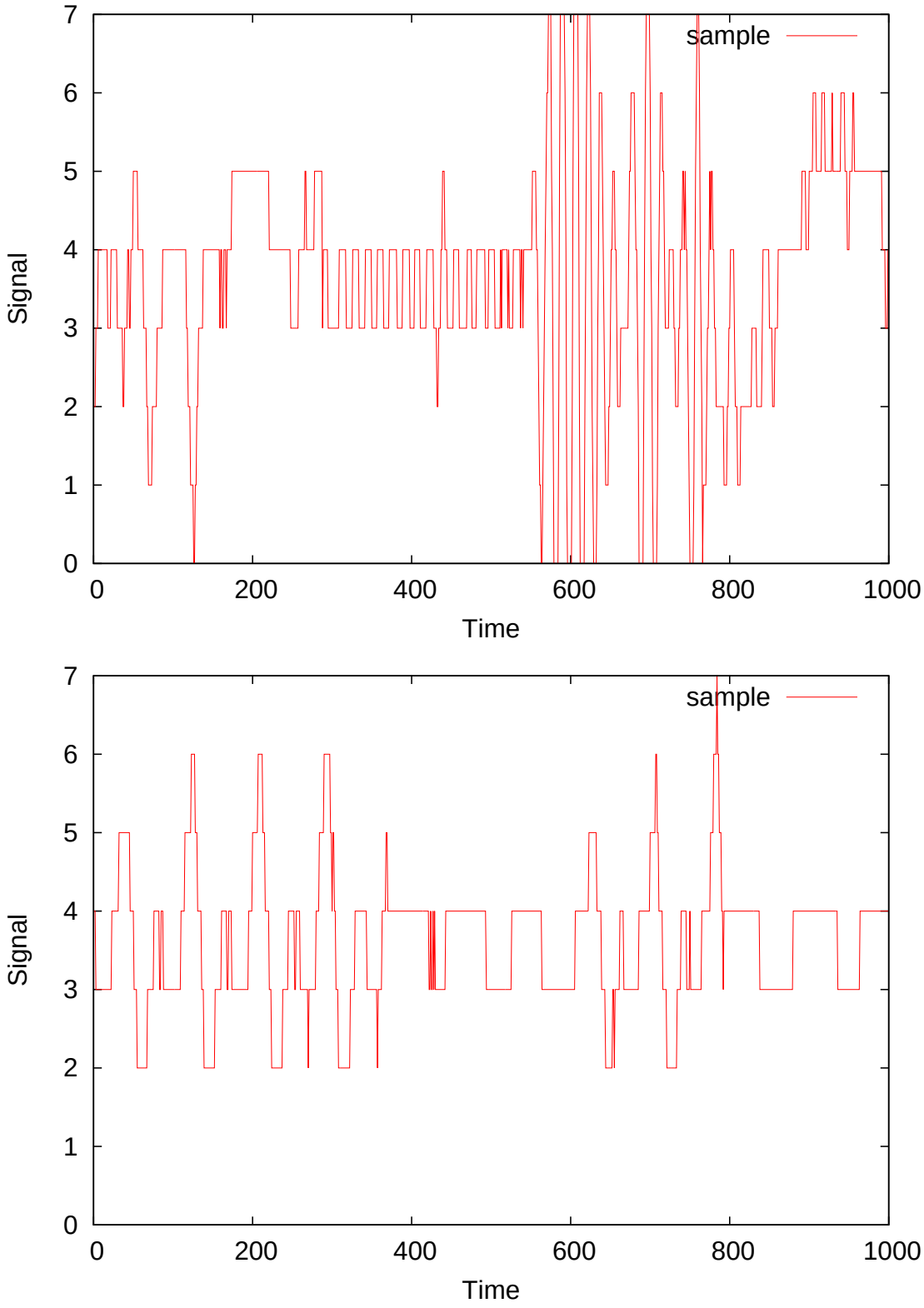

図 3.14. (上：学習データ “k”，下：学習データ “m” の波形の一部．横軸は時間 t，縦軸は Y_t の値をあらわす.

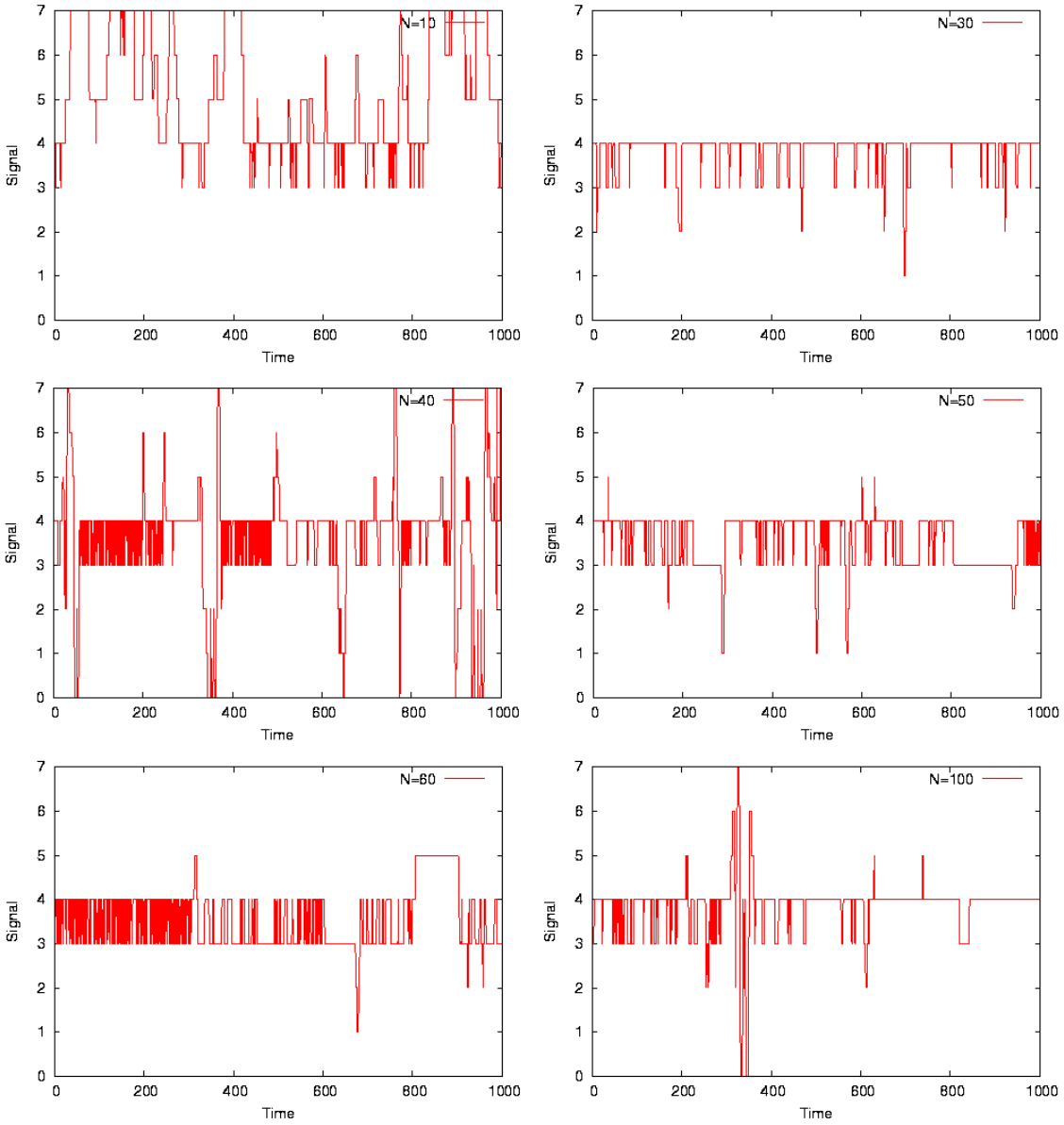

図 3.15. パラメータから生成された [k] の波形のなかの一つ．左から右の順に，上：$N = 10$, 30, 中：$N = 40$, 50, 下：$N = 60$, 100 である．横軸は時間 t, 縦軸は Y_t の値を示している．

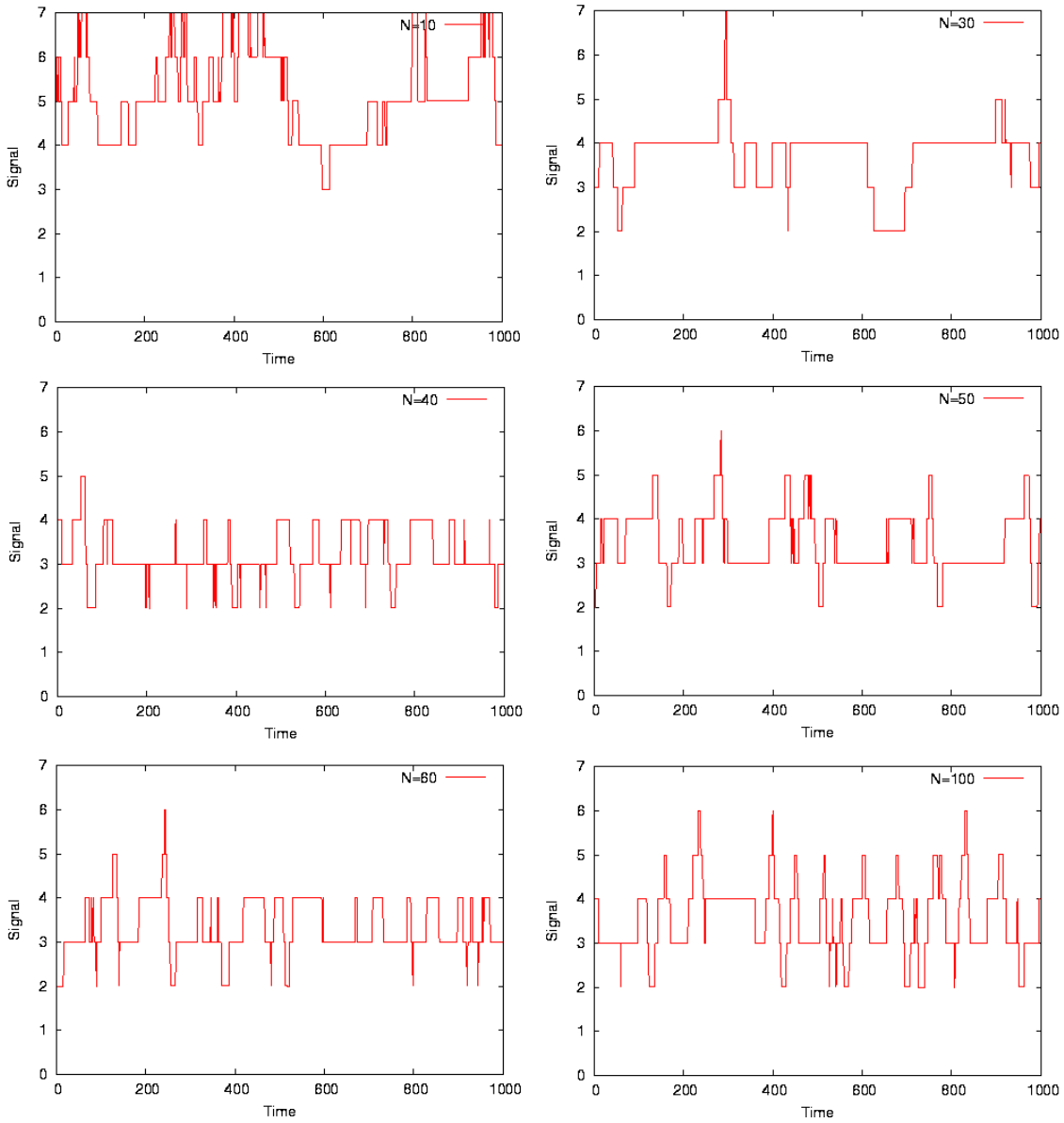

図 3.16. パラメータから生成された [m] の波形のなかの一つ．左から右の順に，上：$N = 10$, 30, 中：$N = 40$, 50, 下：$N = 60$, 100 である．横軸は時間 t, 縦軸は Y_t の値を示している．

第 4 章

まとめ

本研究では，音声データ生成機能を持つ隠れマルコフモデルに現実の音声データを与え，モデルを学習した．学習したモデルから生成された音声信号は，学習信号にはそれほど似ていなかったが，母音の違いや，子音の違いを判別できた．また，隠れマルコフモデルの内部状態の数が多いと，学習データが複雑な確率分布をもっていても学習できることがわかった．今回，X から Y は決定論的に定まる関数を用いた．今後，$P(Y|X)$ のモデル化を工夫することで，より良い音声合成が可能であるかもしれない．また，推定したモデルを用いて認識をし，認識の精度を調べたい．

謝辞

本卒業研究に際して，情報システム工学科，伊達章准教授の多大なご指導を承りました．また，同研究室の 4 回生の 2 人や，修士課程の先輩方とも楽しい時間を過ごさせていただき，楽しく研究をすることができました．そして，他研究室の友人にも論文の添削をしてもらうなど，多くの方々のお世話になりました．宮崎大学で過ごした 4 年間は，多くのことを経験させてもらえた，非常に有意義な時間でした．この場を借りて，これまで関わった皆様に，心から御礼申し上げます．本当にありがとうございました．

参考文献

[1] H. Künsch, S. Geman, and A. Kehagias. “Hidden markov random fields,” The Annals of Applied Probability, Vol.5, No.3, pp.577-602, 1995.

[2] S. Geman and K. Kochanek. “Dynamic programming and the graphical representation of error-correction codes,” IEEE Trans. Information Theory, vol.47, no.2, pp.549-567, Feb. 2001.

[3] 荒木雅弘. フリーソフトでつくる音声認識システム　パターン認識・機械学習の初歩から対話システムまで, pp.2-53, 森北出版株式会社, 2007.

[4] A. Kehagias. “Apporoximation of stochastic process by hidden Markov models PhD thesis”, Division of Applied Mathematics, Brown University, Providence, Rhode Island, U.S.A, Chapter 4, pp.84-110, 1992

付録 A

音声認識

現在，音声認識や音声合成は様々なところで使われている．身近な例を挙げると，音声認識はおもちゃやゲームに使われているし，音声合成は文章読み上げソフトやヴォーカロイドに使われている．ところで，音声認識や音声合成とは何だろう．音声認識とは音声を聞き分ける，つまり，マイクから入力された音をテキスト形式に変換する技術である．そして音声合成はテキスト形式のデータを音として出力することができる技術である．

一般的に音声認識では前処理，特徴抽出，識別の３つのプロセスがある [3]．前処理では入力された音声をコンピュータでの処理や特徴抽出をしやすいデータにする．特徴抽出では認識に必要な情報を取り出す処理を行なう．識別では特徴抽出で取り出されたデータを，様々な音声の見本のデータと比べて近いものを出力する．本論文では特に特徴抽出の部分を取り扱っている．

前処理で行なわれるのは，ディジタル化とノイズの除去である．入力された音声データは連続的なアナログ信号であり，コンピュータが処理できるのは離散的なディジタル信号である．そのため，アナログ信号をディジタル信号に変換する必要がある．この処理には量子化と標本化が使われる．どのくらいのデータを取り出すのかを標本化，データのとる値の範囲を決めるのが量子化である．量子化数・標本化数を大きくすると元の信号を忠実に再現できるが，データ量が多くなり処理が困難になる．逆に小さくするとデータ量は少なくなるが，情報が失われ再現が難しくなる．一般に量子化数は整数のバイト数，標本化数は標本化定理により 16kHZ がよく使われる．そして，入力された音声には周囲の雑音や使用したマイクの性能によって音声のひずみが入ることがある．これをノイズといい，スペクトルサブトラクション法などによって取り除く．

識別は音声データの特徴がどの音に近いのかを調べる．識別に用いる見本のデータをどのような情報にするかが重要になる．特徴抽出部ではこの見本のデータに合わせた特徴を取り出さなければいけない．

特徴抽出では何を識別するのかによって取り出す特徴が違ってくる．今回は音声認識なので，誰が喋ったかに関係なく何を喋っているかに関係のある特徴を取り出さなければならない．音では，振幅が音の大きさを決め，周波数が音の高さや音素を決める．周波数には基本周波数と共振周波数があり，基本周波数が音の高さ，共振周波数が音素を決める．よって，音声

を識別するためには共振周波数が使われることが多い．マイクから入力された音声は時間の関数の波である．周波数の情報を得るためには，周波数の関数の波に変換しなければならない．これをフーリエ変換という．フーリエ変換によって得られた信号をスペクトルという．スペクトルにはいくつかの山があり，この山をフォルマントという．このフォルマントの位置が共振周波数を表している．フォルマントを使うと声の大きさ，高さ，速さに関係なく，同じ音素では近い音が出てくる．しかし，フォルマントには基本周波数の情報も乗っているので，現在の音声認識には MFCC(Mel Frequency Cepstrum Coefficient) という特徴量がよく用いられている．MFCC は複雑なので説明はしないが，簡単に言えばフォルマントから基本周波数を取り除いたようなものである．

付録 B

プログラム

B.0.1 パラメータ推定

```
1  //パラメータ推定
2
3  #include<stdio.h>
4  #include<stdlib.h> /*for srand48(),drand48()*/
5  #include<math.h>   /*for sqrt(),log()*/
6  #include<time.h>   /*srand48()*/
7
8  #define UNITS  2911   //ファイルのデータ数
9  #define STATE  60     //モデルが取りうる状態数 P の order 0...(STATE-1)
10 #define ORI_UNITS 3000  //書き込むデータ数
11 #define Y_VALUE 8   //観測値のとる値 1...Y_VALUE
12 #define ERROR 0.0001  //
13
14 FILE *fp,*fp2;
15
16 int make_Y(char *READFILE,int *y);
17 int make_Ymap(int *x,int *y);
18 double make_X_map(double **p,int *y);
19 double improve_parameter(double **p,double **pre_p,int *y,
20                          double *ini_p,double **L,double **R,double *l);
21 double make_LR(double **p,double *ini_p,int *y,double **L,
22                double **R,double *l);
23 double g(int i,int a);
24 int check_converge(double **gosa);
25
```

```
26  double data_model(int i,int a);
27  double make_prob_function(int *y,int *Ymap);
28  double count_string(int *y,double *bunsi);
29
30  void gnu();
31
32  double *alloc_1d_dbl(int n);
33  double **alloc_2d_dbl(int m,int n);
34  int *alloc_1d_int(int n);
35
36  int main(int argc,char *argv[])
37  {
38    int i,j,k,s,n;
39    int *y,*Ymap;
40    int frag;
41
42    double ave;
43
44    int *Xmap;
45    double **p;
46    double ini_p=0.0;
47
48    y=alloc_1d_int(UNITS);
49    Xmap=alloc_1d_int(ORI_UNITS);
50    Ymap=alloc_1d_int(ORI_UNITS);
51
52    fp=fopen("p-for.dat","w");
53    srand48((unsigned)time(NULL));
54
55    p=alloc_2d_dbl(STATE,STATE);
56
57    //p の初期値
58    for(i=0;i<STATE;i++){
59      for(j=0;j<STATE;j++){
60        p[i][j]=drand48();
61        ini_p+=p[i][j];
62      }
63      for(j=0;j<STATE;j++){
```

```
64          p[i][j]=p[i][j]/ini_p;    //sum_{j}(p[i][j])=1 になるようにする
65        }
66        ini_p=0.0;
67      }
68
69      make_Y(argv[1],y);printf("Read file OK\n");
70      Xmap[0]=make_X_map(p,y);printf("Estimate OK\n");
71
72      for(i=0;i<STATE;i++){
73        for(j=0;j<STATE;j++){
74          fprintf(fp,"%.30f\n",p[i][j]);
75        }
76      }
77      printf("\n");
78      fclose(fp);
79
80      fp2=fopen("model.dat","w");
81
82      for(i=1;i<ORI_UNITS;i++){
83        k=Xmap[i-1];
84        frag=0;
85        while(frag==0){
86          for(j=0;j<STATE;j++){
87            if(drand48()<p[k][j]){
88              Xmap[i]=j;
89              frag=1;
90            }
91          }
92        }
93      }
94
95      make_Ymap(Xmap,Ymap);printf("Make model OK\n");
96
97      ave=Y_VALUE/2;
98
99      for(i=0;i<ORI_UNITS;i++){
100       fprintf(fp2,"%d\n",Ymap[i]);
101     }
```

```
102
103     free(y);free(Xmap);free(Ymap);free(p);
104
105     fclose(fp2);
106     return 0;
107   }
108
109
110   int make_Y(char *READFILE,int *y)
111   {
112     int i,j,k;
113
114     fp2=fopen(READFILE,"r");
115     for(i=0;i<UNITS;i++){
116
117       fscanf(fp2,"%d",&y[i]);
118
119     }
120
121     fclose(fp2);
122   }
123
124   int make_Ymap(int *x,int *y)
125   {
126     int i;
127
128     for(i=0;i<ORI_UNITS;i++){  //Y の設定
129       y[i]=(int)x[i]%Y_VALUE;
130
131     }
132   }
133
134   double g(int i,int y)
135   {
136     double q;
137     double sigma=0.3;
138     if((abs(i%Y_VALUE))==y){
139       return 1;
```

```
140     }
141     else{
142       return 0;
143     }
144
145   }
146
147
148   double make_X_map(double **p,int *y)
149   {
150     int i,j,k,next;
151     int frag;
152     int converge = 0;
153
154     double **gosa,**pre_p;
155     double *ini_p;
156
157     double **L,**R,*l;
158
159     ini_p=alloc_1d_dbl(STATE);                    //最初に i をとる確率
160     gosa=alloc_2d_dbl(STATE,STATE);
161     pre_p=alloc_2d_dbl(STATE,STATE);
162
163     L=alloc_2d_dbl(UNITS,STATE);
164     R=alloc_2d_dbl(UNITS,STATE);
165     l=alloc_1d_dbl(UNITS);
166
167     for(i=0;i< STATE;i++){
168       ini_p[i]=1.0/(double)STATE;
169       for(j=0;j<STATE;j++){
170         gosa[i][j]=1.0;
171         pre_p[i][j]=p[i][j];
172       }
173     }
174
175     while(converge==0){
176       improve_parameter(p,pre_p,y,ini_p,L,R,l);
177
```

```
178       for(i=0;i<STATE;i++){
179         for(j=0;j<STATE;j++){
180           gosa[i][j]=fabs(pre_p[i][j]-p[i][j]);
181           pre_p[i][j]=p[i][j];
182         }
183       }
184       converge=check_converge(gosa);
185     }
186
187     free(gosa);free(pre_p);
188
189     fp2=fopen("first_p.dat","w");
190     for(i=0;i<STATE;i++){
191       fprintf(fp2,"%lf\n",ini_p[i]);
192     }
193
194     frag=0;
195     while(frag==0){                    //Xmap[0]
196       for(i=0;i<STATE;i++){
197         if(drand48()<ini_p[i]){
198           next=i;
199           frag=1;
200         }
201       }
202     }
203     free(ini_p);
204
205     free(L);free(R);free(l);
206     return next;
207   }
208
209   /*遷移確率を更新*/
210   double improve_parameter(double **p, double **pre_p,int *y,
211                            double *ini_p,double **L,double**R,double *l)
212   {
213     int i,j,t;
214     double bunbo,bunsi;
215     double Z;
```

```
216
217     make_LR(pre_p,ini_p,y,L,R,l);
218
219     for(i=0;i<STATE;i++){
220
221       bunbo=0.0;
222       for(t=0;t<UNITS;t++){
223         bunbo+=L[t][i]*R[t][i];
224       }
225
226       Z=0.0;
227       for(j=0;j<STATE;j++){
228         bunsi=0.0;
229         for(t=0;t<UNITS-1;t++){
230           bunsi+=L[t][i]*pre_p[i][j]*g(j,y[t+1])*R[t+1][j];
231         }
232
233         if(bunbo==0.0){
234           p[i][j]=0.0;
235         }
236         else{
237           p[i][j]=bunsi/bunbo;
238         }
239
240         Z+=p[i][j];
241       }
242
243       for(j=0;j<STATE;j++){
244         if(Z==0){
245           p[i][j]=0.0;
246         }
247         else{
248           p[i][j]=p[i][j]/Z;
249         }
250       }
251     }
252
253     bunbo=0.0;
```

```
254     for(i=0;i<STATE;i++){
255       for(t=0;t<UNITS;t++){
256         bunbo+=L[t][i]*R[t][i];
257       }
258     }
259
260     for(i=0;i<STATE;i++){
261       bunsi=0.0;
262       for(t=0;t<UNITS;t++){
263         bunsi+=L[t][i]*R[t][i];
264       }
265       if(bunbo==0.0){
266         ini_p[i]=0.0;
267       }
268       else{
269         ini_p[i]=bunsi/bunbo;
270       }
271     }
272
273   }
274
275   double make_LR(double **p,double *ini_p,int *y,double **L,double **R,double *l)
276   {
277     int i,j,t;
278     double a,add=0.0;
279
280     for(t=0;t<UNITS;t++){
281       l[t]=0.0;
282       for(i=0;i<STATE;i++){
283         L[t][i]=0.0;
284         R[t][i]=0.0;
285       }
286     }
287
288     //L の計算
289     for(i=0;i<STATE;i++){
290       L[0][i]=ini_p[i]*g(i,y[0]);
291       l[0]+=L[0][i];
```

```
292     }
293
294     for(i=0;i<STATE;i++){
295       L[0][i]=L[0][i]/l[0];
296     }
297
298     for(t=0;t<UNITS-2;t++){
299       for(i=0;i<STATE;i++){
300         a=g(i,y[t+1]);
301         for(j=0;j<STATE;j++){
302           L[t+1][i]+=L[t][j]*p[j][i]*a;
303         }
304         l[t+1]+=L[t+1][i];
305       }
306       for(i=0;i<STATE;i++){
307         L[t+1][i]=L[t+1][i]/l[t+1];
308       }
309     }
310
311     //R の計算
312     for(i=0;i<STATE;i++){
313       R[UNITS-1][i]=1.0;
314     }
315
316     for(t=UNITS-2;t>=0;t--){
317
318       for(i=0;i<STATE;i++){
319         for(j=0;j<STATE;j++){
320           R[t][i]+=R[t+1][j]*p[i][j]*g(j,y[t+1]);
321         }
322         R[t][i]=R[t][i]/l[t];
323       }
324     }
325
326   }
327
328   /*収束のチェック*/
329   int check_converge(double **gosa)
```

```
330  {
331    int i,j;
332    int g=0;
333    double diff=0.0;
334    for(i=0;i<STATE;i++){
335      for(j=0;j<STATE;j++){
336        if(gosa[i][j]<ERROR){
337          g++;
338        }
339      }
340    }
341
342    if(g%20==0){
343      printf("%d clear\n",g);
344    }
345    if(g==STATE*STATE){
346      return 1;
347    }
348    else{
349      return 0;
350    }
351
352  }
353
354  void gnu()
355  {
356    FILE *gp;
357
358    gp=popen("gnuplot -persist","w");
359    fprintf(gp,"set parametric\n");
360    fprintf(gp,"set yrange[0:9]\n");
361    fflush(gp);
362    printf("OK\n");
363    fprintf(gp,"plot 'p-for.dat' u 1:2 w l title 'real', 'p-for.dat' u 1:3 w l title 'mo
364  /*   fprintf(gp,"set terminal postscript\n"); */
365  /*   fprintf(gp,"set output 'X60.eps'\n"); */
366  /*   fprintf(gp,"rep\n"); */
367    fflush(gp);
```

```
368     fclose(gp);
369   }
370
371   double *alloc_1d_dbl(int n)
372   {
373     double *new;
374
375     new=(double *)malloc((unsigned)(n*sizeof(double)));
376     if(new==NULL){
377       printf("ALLOC_1D_DBL : Couldn't allocate array of double\n");
378       return(NULL);
379     }
380     return (new);
381   }
382
383   double **alloc_2d_dbl(int m,int n)
384   {
385     int i;
386     double **new;
387
388     new=(double **)malloc((unsigned)(m*sizeof(double *)));
389     if(new==NULL){
390       printf("ALLOC_2D_DBL : Couldn't allocate array of double ptrs\n");
391       return (NULL);
392     }
393     for(i=0;i<m;i++){
394       new[i]=alloc_1d_dbl(n);
395     }
396     return(new);
397
398   }
399
400   int *alloc_1d_int(int n)
401   {
402     int *new;
403
404     new=(int *)malloc((unsigned)(n*sizeof(int)));
405     if(new==NULL){
```

```
406       printf("ALLOC_1D_INT : Couldn't allocate array of int\n");
407       return(NULL);
408     }
409     return (new);
410   }
```

B.0.2 音声ファイル化

```
1   /*
2     datafile.dat -> wavefile.wav
3   */
4
5   #include <unistd.h>
6   #include <fcntl.h>
7   #include <sys/types.h>
8   #include <sys/ioctl.h>
9   #include <stdlib.h>
10  #include <stdio.h>
11  #include <linux/soundcard.h>
12
13  #define LENGTH 1    /* how many seconds of speech to store */
14  #define RATE 44100   /* the sampling rate */
15  #define SIZE 16      /* sample size: 8 or 16 bits */
16  #define CHANNELS 1  /* 1 = mono 2 = stereo */
17  #define ORI_RATE 10000 /*readfile's sampling rate*/
18  #define ORI_SIZE 3 /*readfile's sanpling size*/
19  #define WRITEFILE "result.wav"
20
21  #define MAX 1000000
22
23  /* this buffer holds the digitized audio */
24
25  signed short *buf,*buf2;
26
27  void header_buffer(int datanum,signed short *buf);
28
29  int main(int argc,char *argv[])
```

```
30  {
31
32    if(argc != 2){
33      fprintf(stderr, "Usage: program <inputfile.dat>\n");
34      exit(1);
35    }
36
37    printf("-> %s\n",WRITEFILE);
38
39    int fd;       /* sound device file descriptor */
40    int arg;      /* argument for ioctl calls */
41    int status;   /* return status of system calls */
42    int i,n;
43    int datanum;
44    double count;
45    char *READFILE;
46
47    FILE *fp;
48
49    READFILE=argv[1];
50
51    /* open sound device */
52    fd = open("/dev/dsp", O_RDWR);
53    if (fd < 0) {
54      perror("open of /dev/dsp failed");
55      exit(1);
56    }
57
58    /* set sampling parameters */
59    arg = SIZE;      /* sample size */
60    status = ioctl(fd, SOUND_PCM_WRITE_BITS, &arg);
61    if (status == -1)
62      perror("SOUND_PCM_WRITE_BITS ioctl failed");
63    if (arg != SIZE)
64      perror("unable to set sample size");
65
66    arg = CHANNELS;  /* mono or stereo */
67    status = ioctl(fd, SOUND_PCM_WRITE_CHANNELS, &arg);
```

```
68    if (status == -1)
69      perror("SOUND_PCM_WRITE_CHANNELS ioctl failed");
70    if (arg != CHANNELS)
71      perror("unable to set number of channels");
72
73    arg = RATE;       /* sampling rate */
74    status = ioctl(fd, SOUND_PCM_WRITE_RATE, &arg);
75    if (status == -1)
76      perror("SOUND_PCM_WRITE_WRITE ioctl failed");
77
78    while (1) { /* loop until Control-C */
79
80      /* ファイルから読み込み */
81      if((fp = fopen(READFILE, "r")) == NULL){
82        fprintf(stderr, "Error: %s could not read.\n", READFILE);
83        exit(1);
84      }
85
86      buf = (signed short *)malloc(sizeof(signed short)*MAX);
87      buf2 = (signed short *)malloc(sizeof(signed short)*MAX);
88
89      for(i=0;i<MAX;i++){
90        if(fscanf(fp,"%d",&buf[i])==EOF){
91          break;
92        }
93      }
94
95      datanum=i;
96      printf("datanum %d\n",datanum);
97
98      //fread(buf,sizeof(signed short),datanum,fp);
99
100     fclose(fp);
101
102     printf("You said:\n");
103     status = write(fd, buf, sizeof(buf)); /* play it back */
104     if (status != sizeof(buf))
105       perror("wrote wrong number of bytes");
```

```
106
107
108       header_buffer(datanum,buf);
109
110       /* wait for playback to complete before recording again */
111       status = ioctl(fd, SOUND_PCM_SYNC, 0);
112       if (status == -1)
113         perror("SOUND_PCM_SYNC ioctl failed");
114
115       break;
116     }
117   }
118
119   void header_buffer(int datanum ,signed short *buf)
120   {
121     unsigned char header_buf[44]; //ヘッダを格納する
122     unsigned long fswrh;  //リフヘッダ以外のファイルサイズ
123     unsigned long fmtchunksize; //fmt チャンクのサイズ
124     unsigned long dataspeed;                //データ速度
125     unsigned short blocksize;   //1 ブロックあたりのバイト数
126     unsigned long datasize;                       //周波数データのバイト数
127     unsigned short fmtid;                         //フォーマット ID
128     unsigned short channels;
129     unsigned long rate;
130     unsigned short size;
131
132     int i,j,k;
133     int count;
134     int all_data;
135     int shift;
136
137     FILE *fp;
138     fp=fopen(WRITEFILE,"wb");
139
140     channels=CHANNELS;
141     rate=RATE;
142     size=SIZE;
143
```

```
144     count=RATE/ORI_RATE;
145
146     all_data=datanum*count;
147
148     fmtchunksize = 16;
149     blocksize = channels * size/8;
150     dataspeed = rate * blocksize;
151     datasize=all_data*blocksize;
152     fswrh = datasize + 44  - 8;
153     fmtid = 1;
154
155     printf("datasize %d %d\n",datasize,sizeof(datasize));
156
157     header_buf[0] = 'R';
158     header_buf[1] = 'I';
159     header_buf[2] = 'F';
160     header_buf[3] = 'F';
161     memcpy(header_buf + 4, &fswrh, sizeof(fswrh));
162     header_buf[8] = 'W';
163     header_buf[9] = 'A';
164     header_buf[10] = 'V';
165     header_buf[11] = 'E';
166     header_buf[12] = 'f';
167     header_buf[13] = 'm';
168     header_buf[14] = 't';
169     header_buf[15] = ' ';
170     memcpy(header_buf + 16, &fmtchunksize, sizeof(fmtchunksize));
171     memcpy(header_buf + 20, &fmtid, sizeof(fmtid));
172     memcpy(header_buf + 22, &channels, sizeof(channels));
173     memcpy(header_buf + 24, &rate, sizeof(rate));
174     memcpy(header_buf + 28, &dataspeed, sizeof(dataspeed));
175     memcpy(header_buf + 32, &blocksize, sizeof(blocksize));
176     memcpy(header_buf + 34, &size, sizeof(size));
177     header_buf[36] = 'd';
178     header_buf[37] = 'a';
179     header_buf[38] = 't';
180     header_buf[39] = 'a';
181     memcpy(header_buf + 40, &datasize, sizeof(datasize));
```

```
182
183     fwrite(header_buf, sizeof(unsigned char), 44, fp);
184
185     count=RATE/ORI_RATE;
186
187     all_data=datanum*count;
188     k=0;
189     shift=SIZE-ORI_SIZE;
190
191     for(i=0;i<datanum;i++){
192       buf[i]=buf[i]<<shift;
193       for(j=0;j<count;j++){
194         buf2[k]=buf[i];
195         k++;
196       }
197     }
198
199     fwrite(buf2,sizeof(signed short),all_data,fp);
200     // fwrite(buf2,sizeof(signed short),datanum,fp);
201     fclose(fp);
202
203   }
```