Internal Use Only (非公開)

002

TR-IT-0078

単位接続方式における音声データベースの分類と削減について

Pruning a speech database for concatenative synthesis

森 哲也, ニック キャンベル

Tetsuya Mori, Nick Campbell

1994.9

ABSTRACT

本報告では、音声データベースの削減手法として、音声の基本単位である母音における、その分類方法と削減方法について述べている。母音に対してデータを削減することの有効性と、また分類と削減を行なう情報としてホルマント周波数と帯域幅、そして韻律情報を用いてデータを評価することで分類と削減が行なえることが確認された。そして、母音という基本単位を用いることによってのデータベースのデータ量が効率良く削減されること、また合成音声の劣化も最小限に抑えられることが実際に確認された。

©ATR Interpreting Telecommunications Research Laboratories.

©ATR 音声翻訳通信研究所

目次

1　はじめに

計算機環境がひと昔前に比べて飛躍的に向上し、蓄積できるデータベースの音声データも非常に大きくなった。

音声合成においては合成音の品質向上のため、音声単位としてCV単位よりも大きな単位であるVCV単位、CVC単位等が考えらてきたが、しかし音声単位が長くなるにつれて、データとして蓄積しなければならない音声データも非常に増加した。このように音声単位を長くできるようになったのも、計算機環境の向上があったからに他ならない。

現在、ATRにおいては非均一な音声単位が用いられており、合成音声の品質向上の面においては飛躍的に向上したものの、その分データベースとしての音声データも従来の均一単位に比べて非常に大きなものとなった。しかし、蓄積できる音声データがいくら大きくなったといっても実際には限界がある。

データベースとして蓄積されている非均一音声単位データにおいて、合成音声の品質面からみても削除が可能なデータというものは存在するため、データベースを効率的に構築してそれらを削減することが考えられている。

効率の良いデータベースを構築するにあたって、音声単位としての枠においてこのデータの削減を行なう試みが従来から行なわれているが、このレポートにおいては母音、子音という音声の基本単位においてデータの分類を行ない、データベースの削減手法を考える。

母音は前後の音素の影響が大きく、また韻律に対しての影響が大きいので非常に発話におけるバリエーションが多いと言える。それに比べて、子音は前後の音素の影響が少なくバリエーションが非常に少ない。しかし、従来の削減手法においてはこのバリエーションの多い母音とバリエーションの少ない子音自体がデータの削減に対して同じ次元で考えられ、効率の点においてあまり良い削減方法とは思われない。

よって、母音自体で特徴量を取り出すことができ、その特徴量をもとに母音に対しての分類ができるならば効率の良い分類が可能であると思われる。

また、基本単位という小さな枠組において基本周波数(F_0)、継続時間長などの韻律的要素の影響を見ることにより細かな分類が可能であると思われる。

音素を特徴づける優勢な周波数成分は、口腔や鼻腔など音声器官の共鳴現象に対応しておりホルマントと呼ばれる。ホルマントは、前後の音素の影響も含まれており、また第1ホルマント周波数(F_1)に対してF_0の影響が見られること、また継続時間長が短いほど前後の音素の影響が大きくなる事から分かるように韻律的特徴についても含まれている。

このようにホルマント情報には重要な情報を含んでいることは以前より指摘されているが、ホルマント周波数がスペクトルから安定に取り出すことが難しいという理由から、ホルマント周波数を用いるには細心の注意が必要である。

しかし、測定誤差が非常に大きいことは考慮しなければならないが、第1ホルマントから第4ホルマントの周波数と帯域幅、そして韻律と用いるデータ量を多くすることによって、情報全体の割合からすれば誤差はそれほど大きな割合を占めないと思われる。

よって、母音に対してこのホルマントを考慮した分類を行なうことによって、合成音声の劣化を抑えた効率の高いデータベースの削減が行なえると考え、この考えに基づいて実際に削減を行なった。

本編においては、ホルマントを用いた母音に対する音声データベースの分類方法と削減方法について述べ、またそれぞれの結果について示す。

2　データベース分類・削減の概略

データベースの分類・削減方法について述べる前に、それぞれの手法の概略を述べる。

まず、データベースに含まれるすべての母音に対してホルマント情報を考慮して5母音の分類を行った。(図 1. 参照)

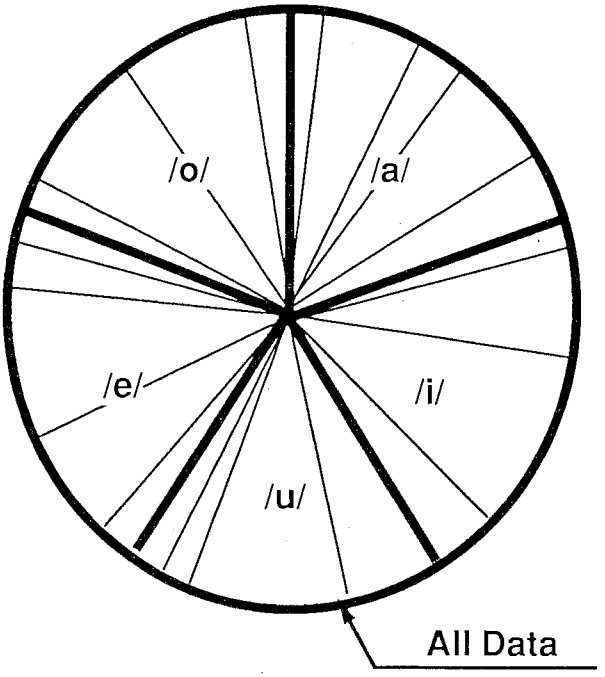


図 1: データベースの分類

図に示すように、分類に対しては5母音すべてを対象のデータとしてその中から分類を行なった。母音 /a/ なら /a/ のなかで分類を行なわなかった理由として、同じ /a/ においても、ある /a/ は /i/ に近いような /a/ であるかも知れないし、同様に母音 /i/ においても、ある /i/ は /a/ に近いような /i/ であると考えられた為である。

そして、分類されたカテゴリーにおいて代表的な母音を選び、そのカテゴリー内における母音はすべてその代表的な母音に置き換えることによって、データベースの削減を行なうことができる。(図 2. 参照)

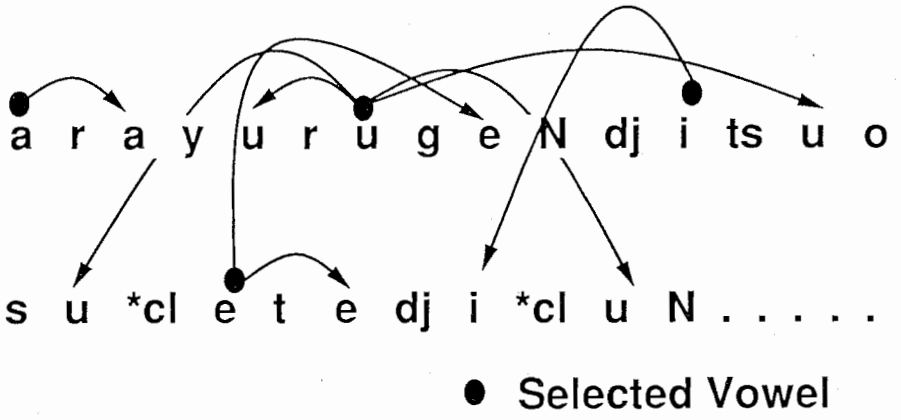


図 2: 母音の置き換え

ここで、図において母音の上に ・ が付いているものを、その分類されたカテゴリーにおける代表的な母音と仮定した場合に、同じカテゴリー内の各母音を置き換える様子を表している。

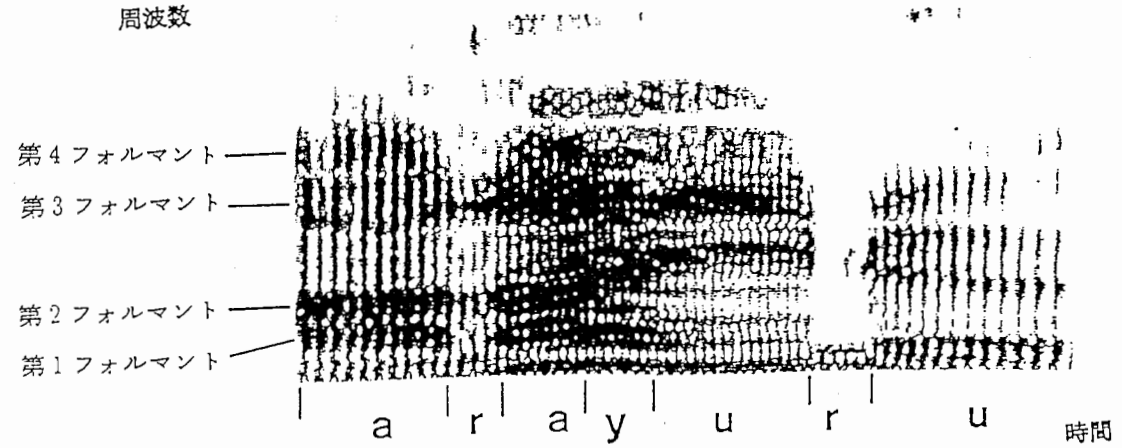
今回は、時間の都合上代表的な母音を選ぶことが出来なかったため、分類されたカテゴリーにおいて各母音を入れ換えることによって、この分類・削減方法の検証を行なった。

以降に、データベースの分類、データベースの削減について詳しく説明を行なう。

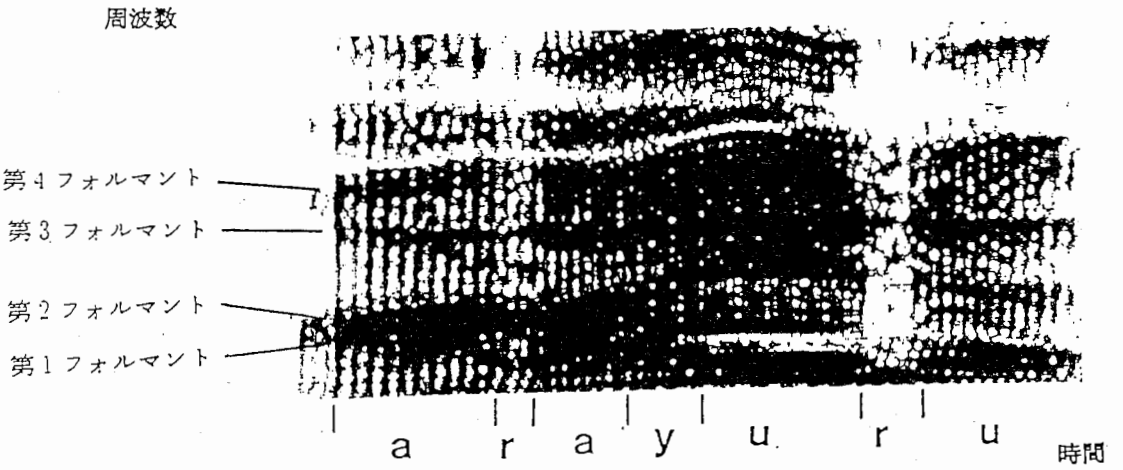
3　データベースの分類について

3.1　ホルマントについて

ホルマント周波数は、第1、第2ホルマントにおいてほぼその音響的特徴を表現することができる。また、高次のホルマントは、母音の種類によって変動することは少なく声道長などに対応して発話者に固有の値を持つことが多いと言われている。



話者 MTK



話者 MYI

図 3: 話者 MTK と MYI のスペクトログラム（「あらゆる....」）

図 3. に二人の男性話者のスペクトログラムを示す。実際、話者 MTK と話者 MYI のホルマントを比べてみると同じ母音 /a/ においても高次のホルマントにいくにつれて、

周波数にかなりのズレがあることがわかる。

また、音素と音素の境界付近(過渡部)において音響的性質が連続的に推移している調音結合の影響も両話者のスペクトログラムにおいて良く確認できる。

この図からもわかるように、ホルマント周波数の変化を見ることによって母音に対する音響的特徴を細かく見ることが可能である。

しかし今回の母音の分類では、ホルマント帯域幅の値も必要であると考えた。

なぜなら、実際には帯域幅の変化が聴覚上に与える影響は比較的少ないと言われているが、男声と女声と比較をすると女声の方が帯域幅が広いことが確認されており、女声の方が F_0 が高いことは周知の通りであり F_0 と帯域幅において何らかの関係があることが考えられためである。

よって、母音の分類においてホルマント帯域幅を考慮することは非常に有用であると言える。

実際に、ホルマント周波数と帯域幅がどのような値を示すのか確認するために、第1ホルマントから第4ホルマントにおいて各値を調べてみた。

表 1. に各ホルマント周波数と帯域幅の最小値、表 2. に最大値、また平均値と標準偏差を表 3. に示す。

なお、データは 503 文データ話者 MTK であり、付録 A にあげる ESPS によって 12 次の LPC によって 10 ms 間隔で求めたものである。

最小値、最大値に抽出ミスらしき値が見られるが、平均値や偏差の値を見ると抽出ミスの数はそれほど多くないと思われる。

表 1: 各ホルマント情報における最小値 [Hz]

		第 1 ホルマント	第 2 ホルマント	第 3 ホルマント	第 4 ホルマント
周波数 (F)	/a/	110.25	742.47	1395.01	2300.40
	/i/	182.89	804.84	1919.39	2369.29
	/u/	81.20	756.37	1915.58	2356.01
	/e/	164.12	1259.81	2016.36	2749.98
	/o/	177.27	488.68	1943.36	2463.25
帯域幅 (Bw)	/a/	35.25	23.00	39.68	38.41
	/i/	15.57	31.29	45.30	29.56
	/u/	25.54	21.97	44.53	31.88
	/e/	36.42	25.65	43.66	26.18
	/o/	35.42	25.65	43.66	26.18

表 2: 各ホルマント情報における最大値 [Hz]

		第 1 ホルマント	第 2 ホルマント	第 3 ホルマント	第 4 ホルマント
周波数 (F)	/a/	1492.75	2781.16	3792.31	4631.00
	/i/	1365.72	3166.48	3538.09	4632.31
	/u/	1270.00	2326.84	3601.38	4823.00
	/e/	787.94	2257.91	3594.60	4603.79
	/o/	1397.43	2642.61	3707.15	4417.70
帯域幅 (Bw)	/a/	2133.89	980.42	3792.31	4631.00
	/i/	1365.72	3166.48	3538.09	4632.31
	/u/	1270.00	2326.84	3601.38	4823.00
	/e/	787.94	2257.91	3594.60	4603.79
	/o/	1397.43	2642.61	3707.15	4417.70

表 3: 各ホルマント情報における平均値 [Hz] と標準偏差 (()の値)

		第 1 ホルマント	第 2 ホルマント	第 3 ホルマント	第 4 ホルマント
周波数 (F)	/a/	625.40(167.91)	1440.38(192.86)	2640.08(270.29)	3716.74(398.39)
	/i/	292.07(73.42)	2029.35(152.64)	2851.24(192.51)	3634.83(398.26)
	/u/	310.93(57.17)	1464.71(243.62)	2315.49(217.95)	3481.05(453.18)
	/e/	402.23(78.86)	1810.89(176.53)	2708.16(266.75)	3676.62(395.30)
	/o/	268.45(68.22)	1140.15(233.84)	2489.54(306.72)	3165.00(369.83)
	all	368.45(171.53)	1140.15(101.59)	2489.54(141.57)	3165.00(140.56)
帯域幅 (Bw)	/a/	375.91(184.76)	158.16(92.21)	286.23(120.45)	277.88(120.74)
	/i/	113.80(103.16)	179.36(139.47)	223.46(126.91)	277.13(134.38)
	/u/	96.35(58.90)	155.45(93.61)	218.58(186.68)	244.43(155.96)
	/e/	183.99(128.92)	188.17(94.10)	287.42(133.77)	253.08(136.47)
	/o/	200.97(115.95)	128.39(75.04)	262.12(136.53)	235.25(123.22)
	all	217.70(170.04)	158.93(370.36)	259.45(309.70)	259.06(407.00)

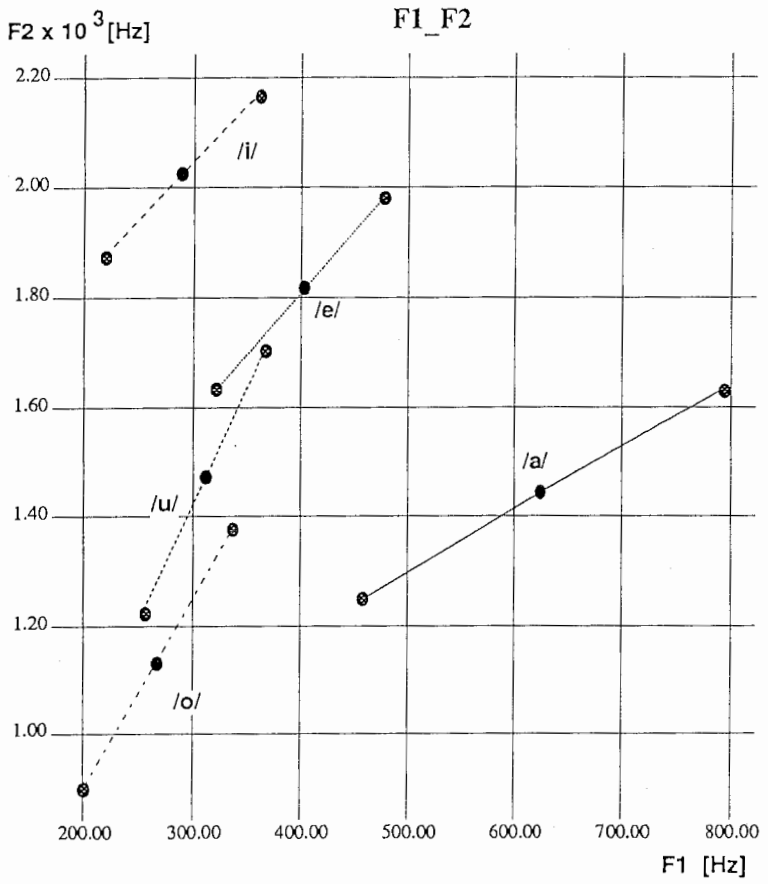


図 4: 話者 MTK における、F_1、F_2 の平均値と標準偏差

これら表に示したデータでは、ホルマントで母音が分類できるかどうかは分かりにくい。

しかし、F_1-F_2 の周波数が母音の音響的特徴を持っており、実際 F_1-F_2 の分布によって母音の分類はおおよそ可能であることが分かっている。

よってその事を確認するため、F_1-F_2 の周波数の分布を実際に求め、それを図 4. に示す。

図において F_1 と F_2 の値を用いることによって、各母音のおおよその分類が可能であることが確かめられた。

3.2 データベースの分類に対する手法

母音の分類に対して、以下の方法を行なった。

1. ホルマント情報 (周波数・帯域幅) を求める。

2. ホルマントをもとに、ベクトル量子化 (VQ) を行なう。

3. 各母音における コードベクトル番号のヒストグラムを求める。

4. ヒストグラムを入力、その母音を出力として Tree Model による母音の分類を行なう。

1 〜 4 の各方法については、以降で詳しく述べる。

なお、今回使用したデータベースは先のホルマント情報についてのデータと同様 ATR503 文データベース Bset、話者 MTK である。

3.3 ホルマント抽出について

ホルマント情報として、第1ホルマントから第4ホルマントまでの周波数($F_1 \sim F_4$)と帯域幅($Bw_1 \sim Bw_4$)を **3.1** にのべたようESPSによって12次のLPCによって10 ms間隔で求めた。(以降、これをホルマント情報と呼ぶ)

そして次に、ラベルファイルの第2層目より各母音の継続時間長を取り出し、継続時間分のホルマント情報の切り出しを行なった。(なお、ラベルファイルの第2層目については付録A参照)

ここで切り出されたホルマント情報は、図5.の様になった。(各値は[Hz])

	第1	第2	第3	第4
Record 1:				
F:				
0:	193.4931	1214.3156	2227.3372	2770.0105
Bw:				
0:	41.012975	881.20793	340.3815	398.33502
Record 2:				
F:				
0:	229.64069	1112.3858	2157.7347	2582.0761
Bw:				
0:	23.606476	461.21467	174.82055	724.43186
		・		
		・		
		・		
		・		
Record 12:				
F:				
0:	311.89641	1103.6449	2139.4149	3767.8567
Bw:				
0:	66.750686	46.606457	271.36072	78.597637

図5: 抽出されたホルマントの例(「あらゆる....」の文頭の /a/)

なお、この抽出されたホルマント情報をみると第1ホルマントの周波数が抽出ミスであると思われるが、これは抽出ミスがみられた悪い例であり、**3.1** のホルマント情報の各値を見れば分かるようそれほど抽出ミスが多くはないと思われたので、ESPSで求められたホルマント情報をそのまま用いることにした。

表 4: 母音の数と切り出されたホルマント情報の数

母音	母音の数	ホルマント情報の数 (レコードの総数)
a	3479	32983
i	2195	19056
u	1789	14297
e	1813	18737
o	2914	31392
合計	12190	116465

また、データベースにおいて発話された母音の数と切り出されたホルマント情報の数は、表 4. に示すようになった。(図.5 に示したようなホルマント情報のレコードの総数)

削減を行なうべきデータ量としては各母音の総数は十分な数であることが、この結果によって確認できた。

3.4 ベクトル量子化について

求められたホルマント情報に対して第 1 〜第 4 ホルマントまでを別々に考えると、非常に複雑で膨大な情報となってしまうので、これらをベクトル量としてとらえて、情報圧縮の手法であるベクトル量子化 (VQ) を用いてこのベクトル量を一つの符号で表し、情報を圧縮することを行なった。

つまり、一つのコードベクトル番号がそのホルマント情報の特徴量を表しているものと考えた。

実際には、ホルマント情報 ($F_1 \sim F_4$、$Bw_1 \sim Bw_4$) を図 5. に示すような 8 次元のベクトルとしてとらえて、その一つ一つにコードベクトル番号を割り当てることを行なった。

VQ コードブックの作成に対しては、クラスタリングの手法を用いてベクトルパターンのセントロイドの計算を行ない、パターンの蓄積をして作成を行なった。

また今回、VQ コードブックの大きさは、16,32,64,128,256 の 5 通りを用いて、それぞれに対して VQ を行なった。

次に、VQ によって求められたコードベクトル番号より、/a/ /i/ /u/ /e/ /o/ それぞれに対してコードベクトル番号の累積を求めた。結果として各母音に対して、それぞれ特徴のある分布をしていることが分かった。(図 7. 〜図 11. 但し、図はコードブック 128 のとき)

この結果から、VQ によって求められたホルマント情報に対するコードベクトル番号が、各母音に対しておおよそ同じような番号を示していることが分かる。

また、総レコード (116465 Record) においての分布は図 12. の様になった。(但し、図は

コードブック 128 のとき)

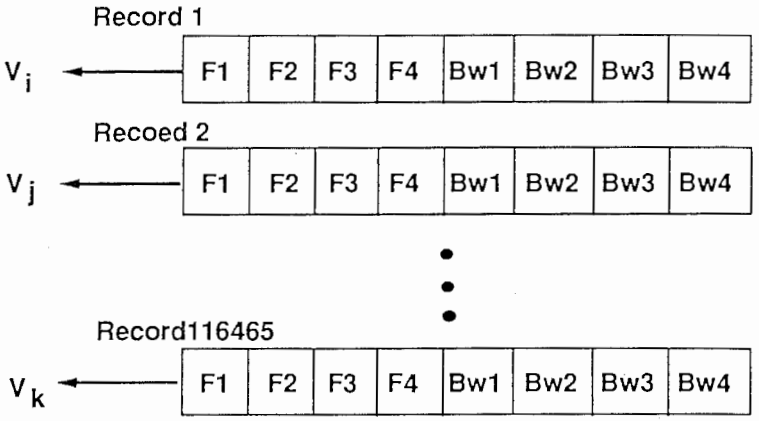


図 6: ホルマント情報のベクトル化

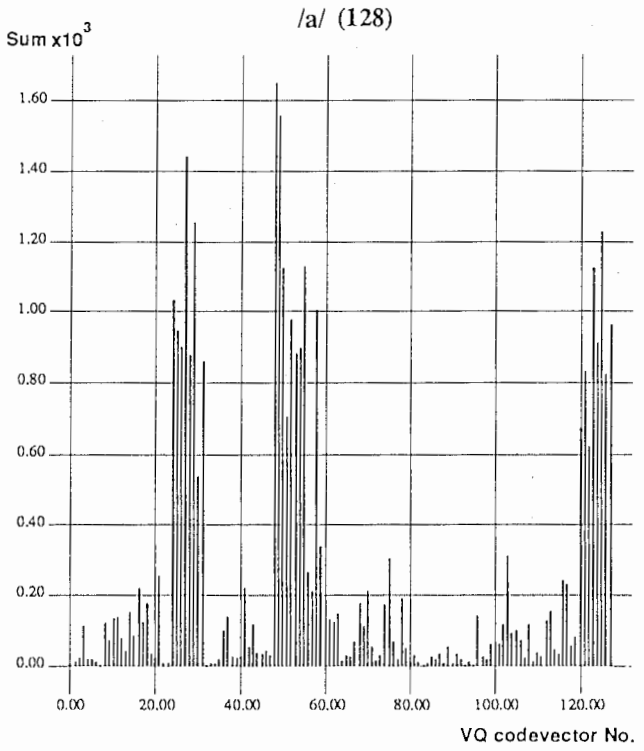


図 7: /a/

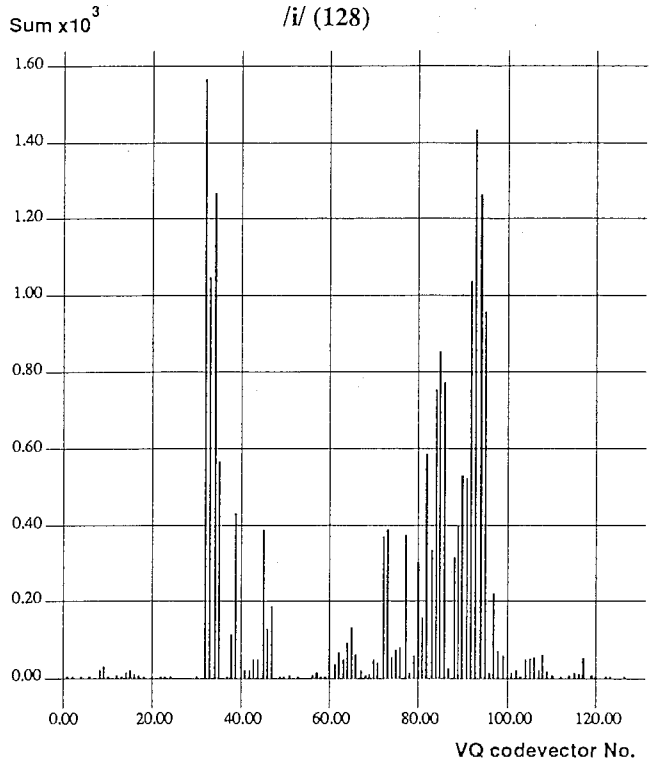

/i/ (128)
Sum x10^3
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.00
20.00
40.00
60.00
80.00
100.00
120.00
VQ codevector No.

図 8: /i/

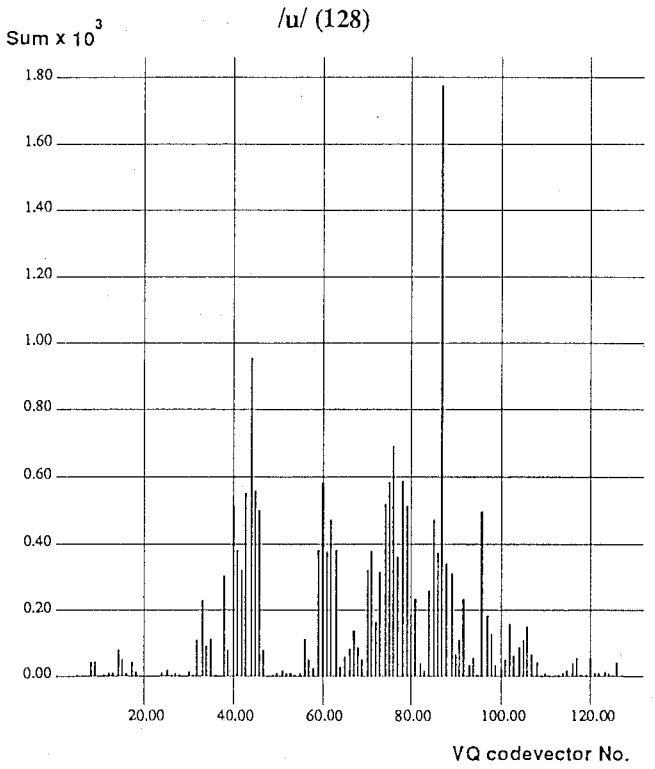

/u/ (128)
Sum x 10^3
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
20.00
40.00
60.00
80.00
100.00
120.00
VQ codevector No.

図 9: /u/

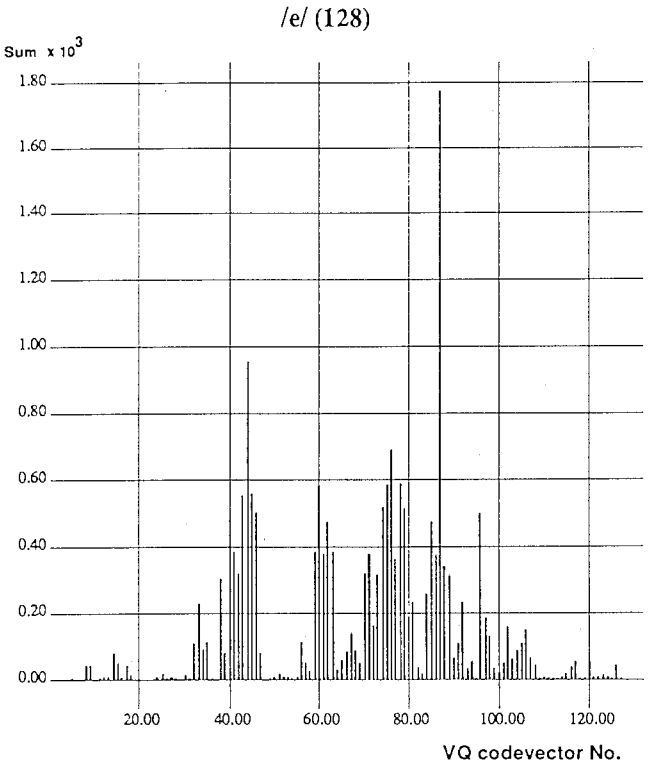

/e/ (128)
Sum x 10^3
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
20.00
40.00
60.00
80.00
100.00
120.00
VQ codevector No.

図 10: /e/

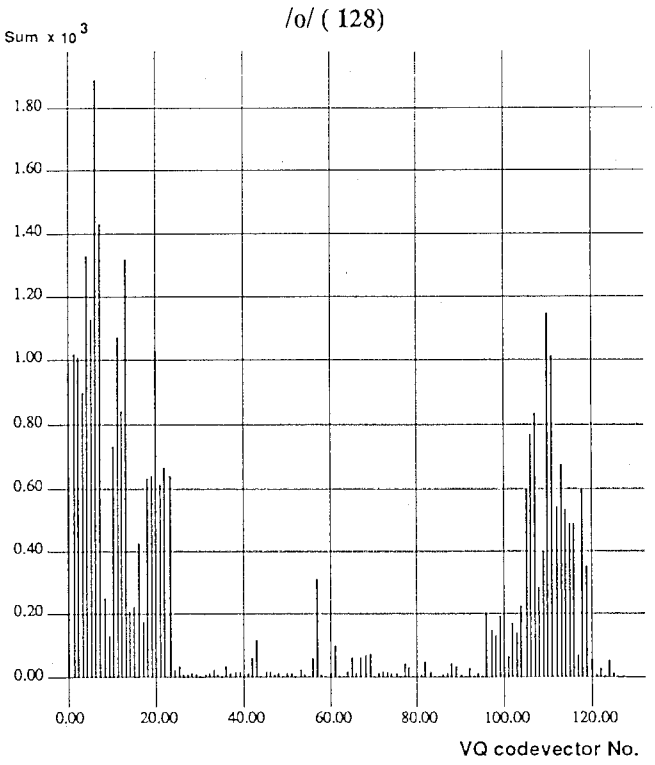

/o/ (128)
Sum x 10^3
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.00
20.00
40.00
60.00
80.00
100.00
120.00
VQ codevector No.

図 11: /o/

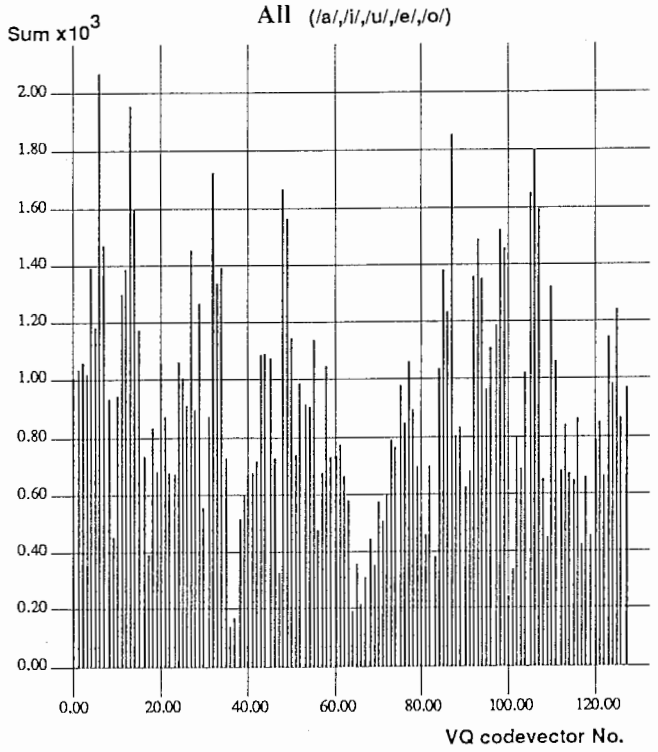


図 12: All(/a/,/i/,/u/,/e/,/o/)

この結果から、それぞれのコードベクトル番号がほぼ偏りなく各レコードにおけるホルマント情報に対して使われていることが分かる。

以上の結果より、VQを用いることによって膨大な情報であったホルマント情報を、非常に小さな情報に変換することができた。

そして、変換された情報が各母音に対する母音性を含んでいることが確認できた。

3.5 VQ コードベクトル番号の正規化ついて

一つのVQ コードベクトル番号そのものでは1レコード分の情報しか含まれていないので、各母音に対するレコード分の情報を求めるために、VQ コードベクトル番号のヒストグラムを求める。ここで、レコードの大きさ(継続時間長の違い)によってデータ(コードベクトル番号)の総数が異なってしまうため、次のように総数(S)を一定にした。

$$S_{v_i} = \sum_{k=1}^{R} V_i, \quad i = 1, ..., 128$$

$$S = \sum_{i=1}^{128} S_{v_i}/R$$

($V_i : VQ\, codevectornumber$, R : 1 母音に対するレコード数)

このようにして求めた、ある /a/ に対するヒストグラムの一例を図 13. に示す。

ヒストグラムを用いることによって、各フレーム毎ではなく一つの母音に対してのコードベクトルの分布が分かるが、但しヒストグラムを用いることによって各フレームに対しての時間情報が無くなってしまい、コードベクトルが時間軸に対して全く反対のデータがあっても、それはヒストグラム上では全く同じものとなってしまうので問題がある。

この問題に対しては、そのようなデータがもしかなりの割合でみつけられるようであれば、改善が必要である。

3.6 データの分類方法について

3.5 によって求められた各母音に対する VQ コードベクトル番号のヒストグラムが求められた。

そして、この求められた各母音のヒストグラムすべてを分類することによって、母音の分類を行なうことが出来る。

分類は、ヒストグラムにおけるデータの分布の割合によって行なうことで行なうことで可能であり、またその条件によってデータの分類を行なわなければならない。

今回は分類された結果に対しては各々の母音が推定されるものとして、入力をヒストグラム、出力を母音として Tree を用いて分類を行なった。

ここで、まず Tree の説明を行なうことにする。

Tree は、データに対して分割を行なうことによってデータの特徴が掴めなくなるまで、また分割がなるべく均等になるように次々と 2 分割していくことによって生成されるモデルのことである。

この、Tree の概略図を図 14. に示す。

よって、Tree は Branch を深く伸ばすことによって入力データを分類していくことになるが、それ以上 Branch を伸ばしてもデータを分類することができないようなときには、意味のないデータの分割を行なってしまうことになる。にする。そこで、Tree は意味のない分割を避けるために Branch を切ることによって (**Pruning**) Branch をそれ以上伸ばしていかないようにする。(図 15. 参照)

よって、Pruning を行なうことによって Leaf そのものの数が減ることになり、データの分類という点からみるとカテゴリーの数が少なくなることになる。しかし、Pruning が多く行なわれると実際にはその Branch でもう少しデータを細かく分析を行なった方が

表 5: テストデータ

出力	入力			
母音	V1	V2	V3	V4
a	0	2	2	1
i	0	5	0	0
u	2	0	2	1
e	0	3	1	1
o	0	0	5	0

良かった場合に対しては、出力との不一致を起こしやすいことになる。また、これはTree自体のアルゴリズムの善し悪しによって決まり不一致の数は多少増減する。

そして、Pruningされず深く伸びていったBranchで、それ以上データの特徴が掴めない場合にはPruningされたBranchと同様にTreeのLeaf (出力) となる。

以上の様にしてTreeは作成されるが、Treeを用いることによってデータの示す隠れた特性あるいは他のデータとの類似点が見つけ出される事になる。

Tree自体の分類方法を説明するために、表 5. に示す簡単なモデルを考える。

このような入力に対して、Treeは図.16 の様に分類を行なう。

まず、最初のノード (節) においての枝分けを行なう。

均等にデータの分割を行うこと、かつデータの特徴を掴みやすいところ (データの値の幅があまりなく、値のバラツキがないところ) で枝分けを行なうので、V4の値が注目される。

そして、V4における最小値 (0) と次に小さな値 (1) の平均値を計算し、その値の大小によって枝分けを行なう。

次に2段目の左のノードにおいては、もうV2でしか分けることができないのでそのまま平均値を計算して枝分けを行なう。右のノードにおいては、データの値のバラツキの最も少ないV1の値によって同様に枝分けを行なう。

そして、3段目においても同じように枝分けを行ない、Treeが作成される。

以上の様に、順次枝わけを行なうことによって図 16. に示すようなTreeが作成される。

今回用いたTreeモデルのTreeの作成方法 (Treeモデル) として、

1. 入力データすべてをTree作成用のデータとしてTreeの作成を行なう手法
2. データの分割を行ない、Tree作成用データと評価用のデータとしてデータの分割を行なうことによって、Treeの作成を行なう手法
3. Bayes推定を用いたTreeの作成手法

の3通りの方法がある。

それぞれの方法について、説明を行なう。

1については、すべての入力データをもとにTreeを作成するため、作成されたTreeを用いてTreeを作成した入力データ以外のデータが入力されると、そのデータの「偶然性」によって本来の出力とTreeの出力の不一致が起こってしまう。よって、Treeを作成したデータのみによって評価する場合はよいが、他のデータを用いると信頼性があまりないといえる。

2については、1における問題を多少防ぐ意味で入力データの分割を行なっている。入力データ以外のデータを用いてTreeの出力を見たい場合は1よりも信頼性は高いといえる。今回のTree作成にあたっては入力データをランダムに3分割し、そのうち2つをTree作成データ、残りを評価用のデータとした。このモデルにおいてはPruningが高い割合で行なわれるのでLeafの数そのものが非常に小さいので、データの分類を大まかに行なう時には都合が良い。

3については、Bayes推定を用いることによってデータが入力されるたびにTreeの各Branch、各Leafにおける確率を計算していきPruningを行なうことなく、確率によってデータの分類を行なう方法である。このモデルはPruningを行なわず、各Branchにおける確率によってデータが枝分けされていくので、信頼性の高いTreeの作成が行なえる。しかし、Pruningを行なわないので、Leafの数が他のモデルと比較して非常に膨大なものとなってしまう。本来の出力とTreeの出力の不一致(推定誤り)を少なくしたい場合には最適であるが、データの分類という面では、あまり都合が良くない。ベイズ推定を用いることによって、分類におけるミスがかなりの割合でなくなる。

3.4でも説明を行なったが、今回はVQコードブックの大きさを16,32,64,128,256の5通りでTreeでの分類を行なった。そして、Treeの作成にあたっては先に示した3通りのモデルを用いた。

しかし、VQコードブックの大きさが256のときはTreeの作成において1つの出力に対する入力の値が大きすぎたため、Treeの作成を行なうことができなかったので、Treeの結果としては4種類である。

それぞれのVQコードブックにおける各Treeモデルにおける評価結果は、図.15に示す通りとなった。評価としては、本来の入力に対する出力の値と、実際にTreeで出力された値における不一致(推定誤り)数が全体の入力データ数に占める割合を用いている。

よって、この評価値が(Treeのデータに対する分類の正確度)100[%]に近いほどTreeモデルが良いといえる。

(付録Bにおいて実際のTreeの各出力結果を示す。)

図15.の結果をみると、VQコードブックの大きさが128のときが、どのモデルにおいても分類の正確度が高く、正確度の値はほぼ飽和していることが分かる。

また、各手法においてのTreeモデルの分類の正確さはベイズの手法を用いた場合が非常に高く、分割法においては作成データの不足また、テストデータの作成されたTreeに

表 6: 各母音における出力の不一致の全データにおける割合 [%]

Model	/a/	/i/	/u/	/e/	/o/
入力データすべて	+2.22	+12.06	-4.10	-2.05	-8.12
分割法	-9.02	+15.67	-1.23	-6.89	1.48
Bayes 推定	+4.02	+7.06	+5.00	-10.01	-6.07

表 7: 前後のラベル情報を加えた場合の Trees の分類の正確度

	ヒストグラムのみ	前音素	後音素	前後の音素
入力データすべて	91.03	93.99	92.76	94.63
分割法	87.56	88.42	89.94	89.11
Bayes 推定	96.05	98.82	97.42	98.95

対する偶然性が大きいため、分類に対する正確さが低い (推定誤りが大きい) ことが分かった。

また、それぞれの Tree モデルにおける各母音の全データに対する本来の出力と Tree の出力の不一致数の割合 (推定誤り) を表 6. に示す。(コードブックの大きさが 128 の場合)

ここで、表においては “+” は Tree における実際の出力が本来の出力よりも多く観測された場合の全体に占める割合、 “−” は実際の出力が本来の出力数よりも少ない場合の全体に占める割合を表す。

表 6. の結果より各モデルにおける、母音に対する推定誤りはモデルによってバラツキがあり、これによってどのモデルが最適であるかは判断できなかった。

図 17. の結果より Tree を用いてヒストグラムの情報における母音の分類を高い精度で分類できることが確認できた。

しかし、今用いているヒストグラムにおける情報がもとのホルマント情報を保持しているかどうかは分からないので、次の様な実験を行なった。

ヒストグラムにその母音の使用環境 (前後の音素) の影響が含まれているかを確認するために入力データに対して、ヒストグラムの他に母音の前後のラベル情報をラベルファイルの二層目より切り出し、それを加えて Tree での評価をした。(図 18. 参照 但し、コードブック 128 のとき)

もちろん、ラベルファイルが加えられることによってそれだけ分類に対する正確さが高くなるは当然と考えることができ、表 7. の結果からもそのことが確認できた。

また結果から、どのラベル情報を加えた場合においてもヒストグラムのみを加えた場合に対して、Tree による分類が若干向上していることが分かる。

しかし、一番良い結果が示された前後のラベルを加えた場合においても 2[%] 程度の改

表 8: 母音に対して時間分割を行なった場合に対する Tree の分類の正確度

	R"$_1$	R"$_2$	R"$_3$	R"$_1$+R"$_2$	R"$_2$+R"$_3$
分割法	85.35	88.42	72.40	88.98	88.01
Bayes 推定	88.91	90.98	85.38	93.74	94.64

表 9: 韻律情報を加えた場合の Tree の分類の正確度

	ヒストグラムのみ	F_0 のみ	継続長のみ	両方を加えた場合
分割法	87.56	85.04	86.77	86.05
Bayes 推定	96.05	96.39	96.55	97.39

善であり、ヒストグラムのみの情報でも、前後の音素の影響を考慮した母音の分類ができていると考えられる。

母音のホルマントは図のように時間軸において前の方では前音素の影響を強く受け、後ろの方においては後音素の影響を強く受ける。(調音結合)

この影響の度合を比べるために図 19. に示すようにホルマント情報の分割を行ない、Tree での分類を行なった。

その結果を、表 8. に示す。(但し、コードブック 64 のとき)

この結果から分かるように、時間軸において後ろの方にいくに従って母音のホルマント情報においては、その母音の持っている母音性に対して後続音素の影響がかなり加わってくることが分かる。

よって、ヒストグラムの情報に時間が考慮されていないが、この結果を見ると時間軸において母音の前部と後部とではかなり情報が異なっていると考えることができる。ある程度はヒストグラムに前後の音素の影響が含まれていると考えると、時間的な要素が含まれていると考えることができる。もし、時間的な要素が含まれていれば、先にあげたヒストグラムを用いることによる問題点はある程度解決されているものと思われる。

次に、この結果をもとに韻律情報を入力データに加えた上で Tree でのデータの分類を行なった。

用いる韻律情報は、基本周波数 (F_0) と継続時間長の 2 種類を用いた。

まず、基本周波数を少数点以下 3 桁の精度で、また継続時間長を少数点以下 1 桁の精度で入力データに加えて Tree での分類を行なった。結果を表 9. に示す。(但し、コードブックの大きさが 128 のとき)

表 9. をみると Tree での分類の正確さの点においては、韻律情報はあまり影響がないことが分かる。

しかし、実際に Tree での分類において韻律情報は多少ではあるが用いられていること

がTreeの出力結果からは得られ、特に継続時間長についてはBranchがあまり深くないところにおいて用いられている場合もみられた。

ここで、あまり韻律情報が考慮されなかった理由として、ヒストグラムにおけるデータがもともとVQによって分類されている情報であるのに対し、韻律情報のデータの精度が高いためTreeで分類する際に分類を行なうにはデータが細かすぎたためと思われる。

そこで、韻律情報についてもある程度の分類を前もって行ない、それを入力データとして加え、再度Treeの作成を行なうことにした。

韻律情報を、平均値と標準偏差を用いて図20.のように3つに分類を行なった。

そして、これを入力データに加えてTreeでの分類を行なった結果を表10.に示す。

表10. 分類された韻律情報を用いた場合

	ヒストグラムのみ	分類された韻律情報を加えた場合
入力データすべて	91.02	90.90
分割法	87.56	84.80
Bayes推定	96.05	96.66

表10.の結果を見ると、表9.の結果に比べ若干分類の精度が上がっていることが確認できる。また、実際のTreeの出力結果からF_0の値についてもBranchがあまり深くないところでノードにおける場合分けで用いられる場合もみられた。

結果より、韻律情報をTree作成前にある程度の分類を行い、そのうえでTreeの作成を行った方がTreeの正確度が高く、分類についても誤りが少なくなるといえる。また、韻律情報を考慮した分類が表10.の結果より高い精度で行えることが確認できた。

以上、Treeを用いてヒストグラムの分類を行なったが、結果より、VQコードブックの大きさを128のときのヒストグラムを分類に対して用いることにした。また韻律情報に対してはある程度の分類をTreeで分類する前に行ない、その分類結果を用いてTreeで分類を行なうことにした。

Treeの各モデルについては、分類の精度が高くなるとLeafの数が増えてしまい、効率のよい分類ができなくなるという問題があるので、もう少し結果を詳しく見ることにし、4に述べることとする。

4 データベース削減結果について

実際にTreeでのデータの分類結果をもとに、データベースの削減方法について考える。

いま、Treeの各Leafに出力された値というものはデータの特徴(ヒストグラムの分布の特徴)が似通ったものであるといえる。よって、各Leafに出力された値を1つ、あるいは数種類にまで削減を行なうことによってデータの分類、かつデータの削減が行なえると考えることができる。

TreeにおいてのLeafの数とLeafでの出力を表10.に示す。

しかし、図 21. 図 22. に示すように Leaf での出力は両 Tree モデルにおいて同じような分布をしている。Leaf の出力の大部分は 1,2 個ぐらいのデータしか観測されず、また逆に一つの Leaf に 100 以上ものデータが観測されるところもある。

この結果からでは、実際に Bayes 推定を用いた Tree の作成法においてはどちらのモデルを用いることによって合成音声の劣化が防ぐことができるのかは確認することができなかった。また、2 のデータベースの分類・削減の概略に書いたとおり、今回は時間の都合上分類された結果に対して代表的な母音を決めることはできなかった。そのため、かわりにこの分類された結果がはたして有効であるかどうか確かめるために、図 23. に示すようにデータベースの 1 文において母音の入れ換えを行なってみた。

実際に、データの入れ換えを行なった文と入れ換えなかった元の文とを主観評価で聞き比べた。

なお、入れ換えにおいては任意に入れ換えを行ない、同じ Leaf の出力である /a/ において一つの /a/ を用いて代表させるのではなく複数の /a/ を用いた。

その結果として、各母音の切り出しを行なう際に、切り出すポイントが微妙なためと、ラベルデータにおいての母音の開始時間と終了時間において、前後の音素が若干含まれてしまっているために、前後の音素が切り出された母音に含まれてしまって、そのためにおかしく聞こえた母音以外は、あまり違和感なく聞くことができた。

よって、母音の切り出しをきちんと行ない分類されたカテゴリー内において代表的な母音を用いることによって、今回の実験結果以上の評価が期待できるものと思われる。

Record Number ・・・・ 10

Record	VQ codevector number
1	29
2	30
3	13
4	13
5	13
6	13
7	3
8	10
9	31
10	11

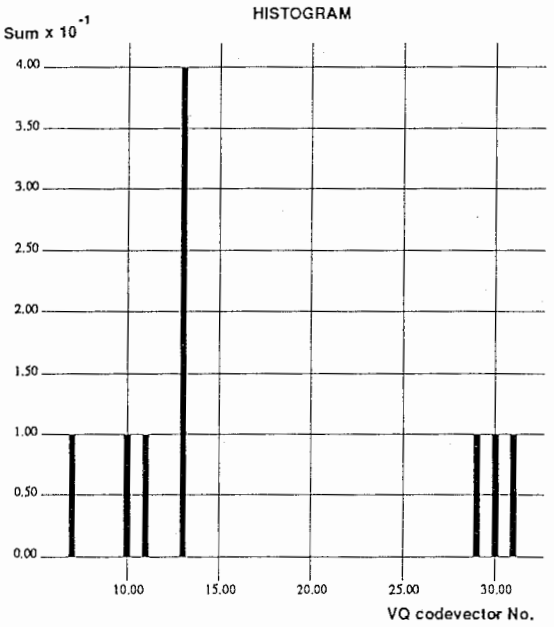


図 13: ある /a/ のヒストグラム

(VQ コードブック 32 の時)

表 10: Leaf における出力

	Leaf の数	Leaf における出力の平均値	最大値	最小値
ベイズ推定	2576	9.46	670	1
入力データすべて	557	21.92	623	2

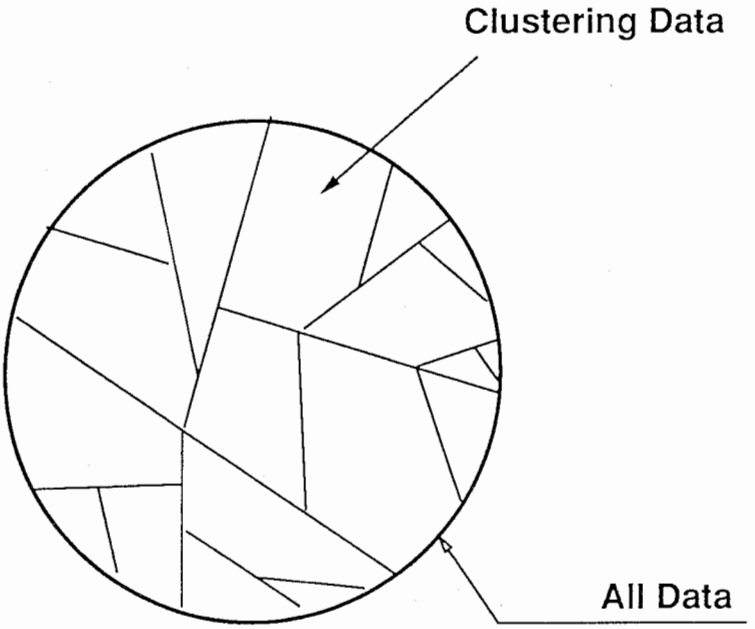


図 14: Tree の概略図

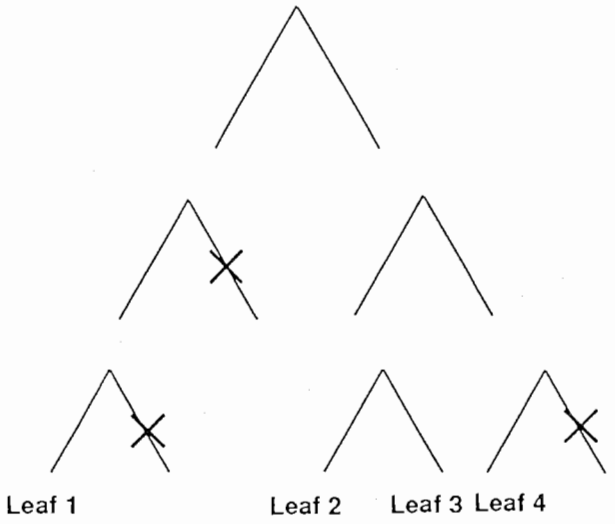


図 15: Tree における Pruning(枝切り)

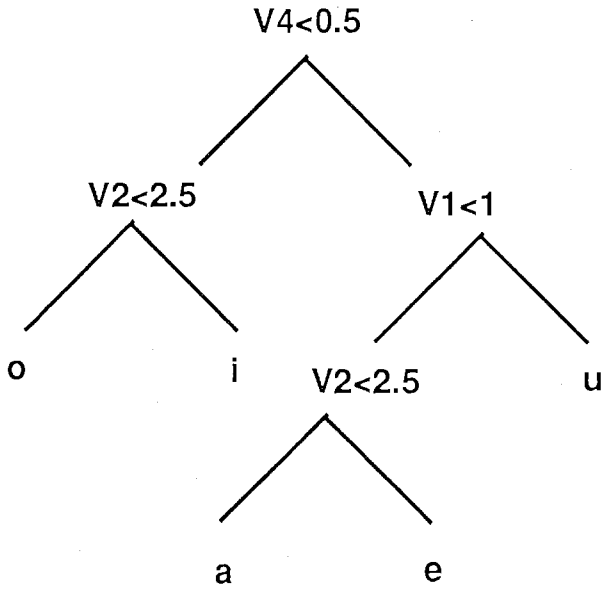


図 16: 表 5. に対して作成された Tree

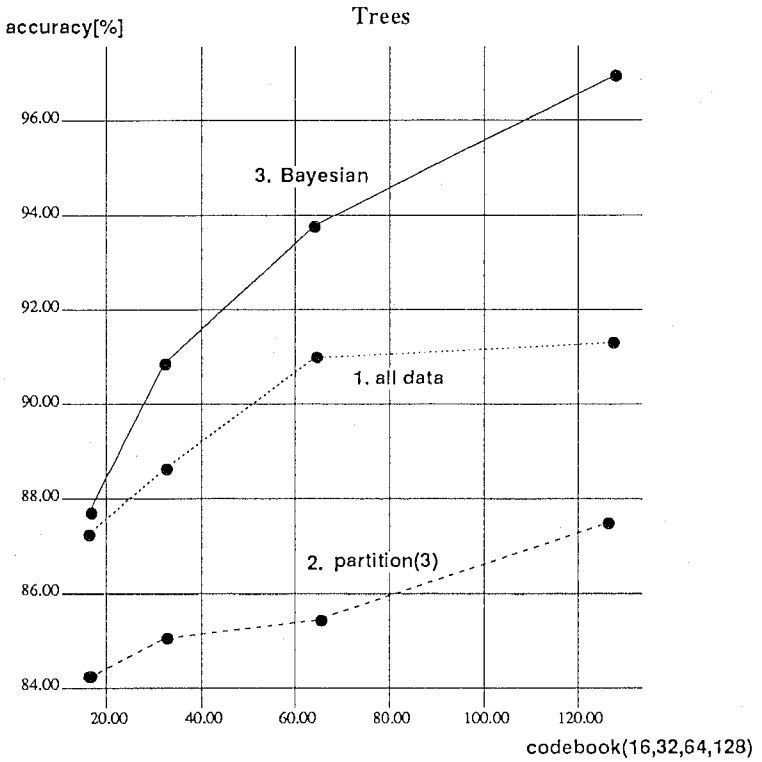


図 17: 各コードブックの大きさにおける Tree の分類の正確度

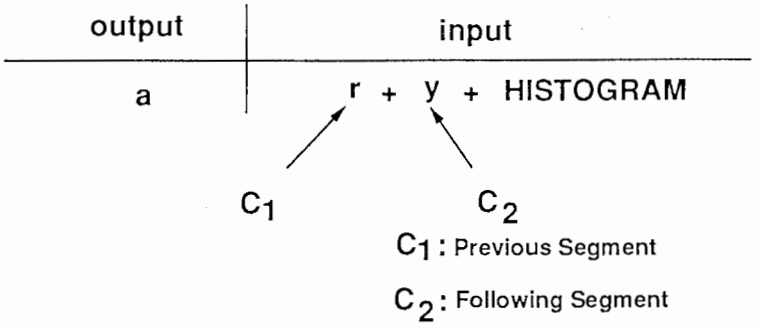


図 18: 加えられた前後のラベル (音素) の情報

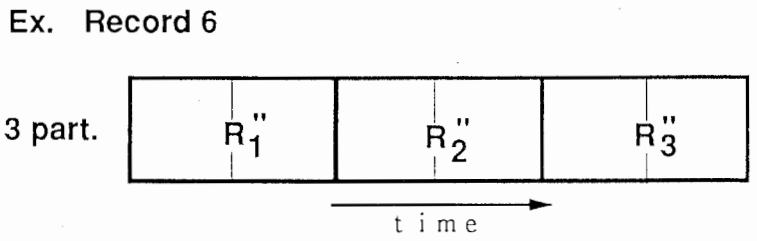


図 19: 時間における分割法

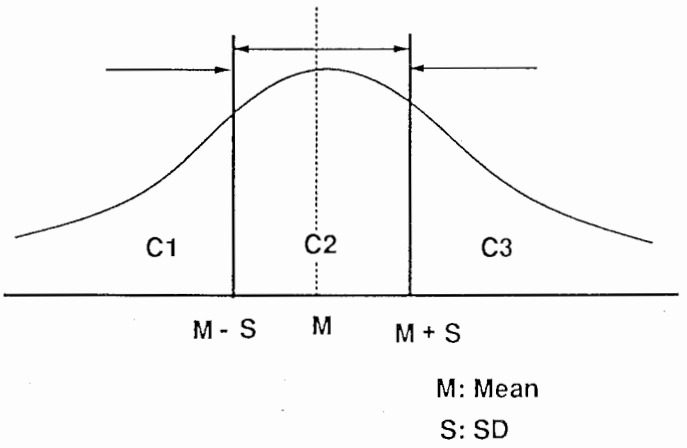


図 20: 韻律情報の分類

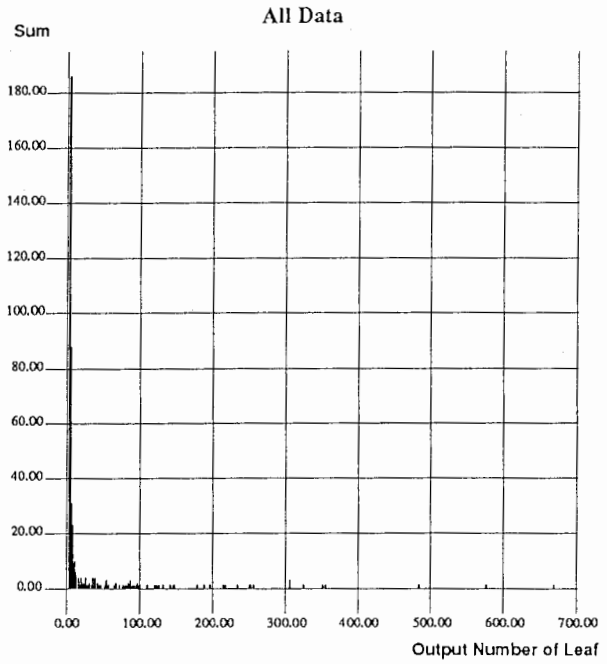


図 21: すべてのデータにおける Leaf の出力

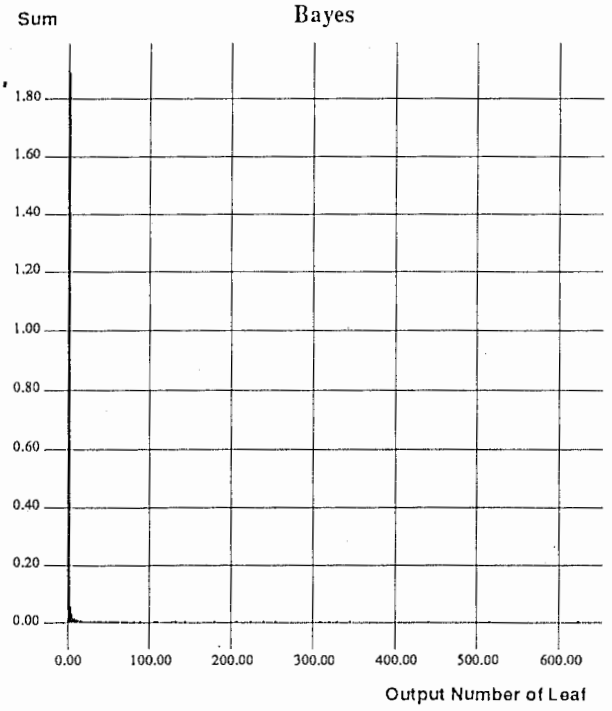


図 22: Bayes 推定における Leaf の出力

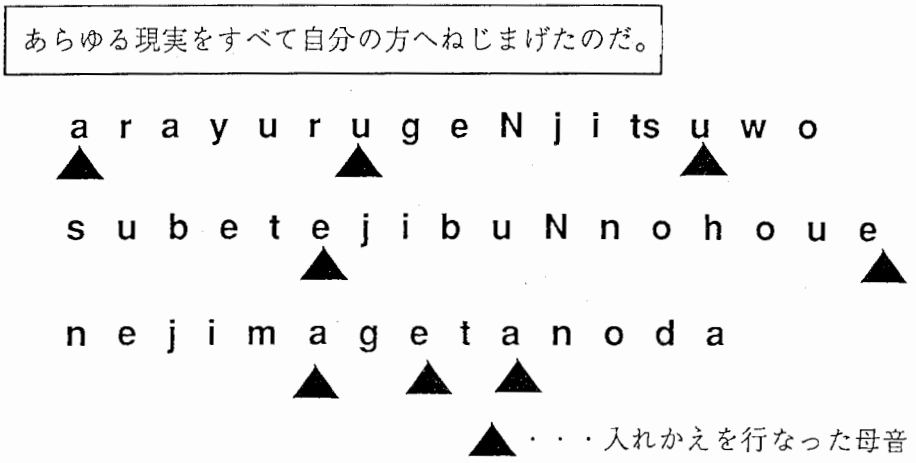


図 23: 母音の入れ換えを行なった文データ

5 考察

今回は、単位音声のうち母音についてのデータベースの分類と削減を行なった。

母音の分類に対しては、ホルマント情報を用いることによって母音に対して細かな分類がなされることがわかった。そして、データベースの分類と削減においては母音で行なうことによって、合成音声の劣化を抑えた分類と削減が行なえることが分かった。

しかし、この報告においては

- 代表的な母音の選択方法
- どの Tree モデルを用いるかの選択
- Tree における出力が実際の母音と不一致している母音に対する対策

については時間の関係上、調べることが出来なかった。

よって、上にあげた問題点を解決するために、実際に合成システムに組み込んで削減された結果について考察する必要がある。

また、今後の課題として以下のことが必要である。

- Leaf の数によるデータの分類の違いがどのようにデータに影響を及ぼすかの検討。
- 分類されたデータにおいて、どの母音を用いればよいかの検討
- 他話者 (女声データ) においての分類と削減を行なうことによる方法の検討
- 子音に対してのデータの分類と削減

以上の事を考慮に入れることにより、さらなる効率の良いデータベースの削減が合成音の劣化を抑えて行なうことができると思われる。

参考文献

[1] Nick Campbell: "韻律を用いた音声合成の単位選択", 日本音響学会平成 6 年度秋季研究発表会講演論文集, 2-5-10,285-286(1994)

[2] Abe, K.,Takeda, K., & Sagisaka, Y.(1989): "On the concatenation of speech synthesis units according to unit extraction context",IEICE JAPAN,SP89-66,17-22

[3] Maria-Gabriella Di Benedetto: "Acoustic and perceptual evidence of a complex relation between F_1 and F_0 in determining vowel height",Journal of Phonetics 22,205-224(1994)

[4] 古井 貞熙: デジタル音声処理, 東海大学出版会 (1985)

[5] ATR 国際電気通信基礎技術研究所 編: 自動翻訳電話, オーム社 (1994)

6 付録

付録 A

- 今回使用したソフトウエアを示す。

1.	ホルマント抽出	ESPS formant (COMMAND)	ENTROPIC RESEARCH LAB.
2.	ホルマント切り出し	ESPS fea_deriv (COMMAND)	ENTROPIC RESEARCH LAB.
3.	Tree Model	mktree (COMMAND)	RIACS & NASA A.R.C.
4.	F_0 抽出	ESPS formant (COMMAND)	ENTROPIC RESEARCH LAB.

- Tree について
  参照
  Statistics and Computing journal: a version with some minor changes appeard in Volume 2,1992,pages 63-73.

- ラベルファイルについて
  2 層目のデータは以下のようになっている。

600	a	1830
1830	r	2040
2040	a	3090
3090	y	3570
3570	u	4470
4470	r	4860
4860	u	6390
6390	*>	6570
↑	↑	↑
開始時間 [ms]	ラベル記号	終了時間 [ms]

なお、ラベル記号については以下の文献を参照のこと。
参照
ATR Technical Report 研究用日本語音声データベースの作成書 (連続音声データ編),TR-I-0086,pages 15

付録 B

Tree の出力結果について

出力結果の見方 (重要なところのみ) について、図 24. に示す。

Parcentage accuracy for tree 1 = 87.xxx ・・・ 出力の正確さ

Leaf count for tree ・・・ Leaf の数

Misclassification matrix for tree 1: ・・・ 以下に説明
(row = ...

実際に観測されたデータに占める各母音の割合

実際に観測されたデータ

	/a/	/i/	/u/	/e/	/o/	All
/a/	0.277276	0.001066	0.001231	0.002625	0.003938	0.286136
/i/	0.000328	xxxx	xxxx	xxxx	xxxx	xxxx
/u/	0.000820	xxxx	xxxx	xxxx	xxxx	xxxx
/e/	0.003035	xxxx	xxxx	xxxx	xxxx	xxxx
/o/	0.003938	xxxx	xxxx	xxxx	xxxx	xxxx
	0.285398	0.180066	0.146760	0.148728	0.239048	1.00000

本来のデータ数（全データ＝1）

全データを 1 とする

図 24: 出力結果の見方について

VQ コードブック 16 のときの Tree の結果について
ベイズ推定の場合

```
Percentage accuracy for tree 1 = 87.6948 +/- 0.297529
Mean square error for tree 1 = 0.186291
Expected accuracy for tree 1 = 85.3253
Typical std. dev. of expected accuracy for an example = 7.61684
Neg. Log Posterior for tree 1 = 5812.7 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 3323, expected = 425.665405
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.277276   0.001066   0.001231   0.002625   0.003938 | 0.286136
  0.000328   0.166448   0.005906   0.025267   0.000410 | 0.198359
  0.000820   0.000574   0.103774   0.005742   0.019606 | 0.130517
  0.003035   0.011321   0.009106   0.114848   0.000492 | 0.138802
  0.003938   0.000656   0.026743   0.000246   0.214602 | 0.246185
-----------------------------------------------------------
  0.285398   0.180066   0.146760   0.148728   0.239048 | 1.00000
```

分割法の場合

```
Percentage accuracy for tree 1 = 84.2084 +/- 0.330285
Mean square error for tree 1 = 0.250026
Expected accuracy for tree 1 = 84.2084
Leaf count for tree 1 = 67, expected = 67.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.272683   0.002543   0.002133   0.006481   0.006399 | 0.290238
  0.000656   0.158983   0.005824   0.027810   0.001148 | 0.194422
  0.001723   0.003363   0.099426   0.009106   0.022724 | 0.136341
  0.004348   0.013700   0.010254   0.103035   0.000820 | 0.132158
  0.005989   0.001477   0.029122   0.002297   0.207957 | 0.246842
-----------------------------------------------------------
  0.285398   0.180066   0.146760   0.148728   0.239048 | 1.00000
```

VQ コードブック 32 の場合
ベイズ推定の場合

```
Percentage accuracy for tree 1 = 90.8532 +/- 0.261098
Mean square error for tree 1 = 0.143602
Expected accuracy for tree 1 = 87.3226
Typical std. dev. of expected accuracy for an example = 8.50854
Neg. Log Posterior for tree 1 = 5362.47 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 3615, expected = 625.291443
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.281952  0.001477  0.001395  0.001723  0.001477 | 0.288023
  0.000082  0.168335  0.004020  0.017637  0.000246 | 0.190320
  0.000328  0.000738  0.117063  0.005250  0.020263 | 0.143642
  0.001313  0.009188  0.008532  0.124118  0.000000 | 0.143150
  0.001723  0.000328  0.015751  0.000000  0.217063 | 0.234865
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

分割法の場合

```
Percentage accuracy for tree 1 = 85.0287 +/- 0.323155
Mean square error for tree 1 = 0.239736
Expected accuracy for tree 1 = 85.0287
Leaf count for tree 1 = 76, expected = 76.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.269729  0.000656  0.001477  0.005414  0.004758 | 0.282034
  0.002133  0.154717  0.006317  0.018868  0.001148 | 0.183183
  0.003035  0.003527  0.101641  0.007301  0.024200 | 0.139705
  0.003610  0.020016  0.015422  0.115915  0.000656 | 0.155619
  0.006891  0.001148  0.021903  0.001231  0.208285 | 0.239459
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

VQ コードブック 64 の場合
ベイズ推定の場合

```
Percentage accuracy for tree 1 = 93.7982 +/- 0.218451
Mean square error for tree 1 = 0.101487
Expected accuracy for tree 1 = 89.2543
Typical std. dev. of expected accuracy for an example = 9.28598
Neg. Log Posterior for tree 1 = 4807.15 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 3320, expected = 791.019409
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.283757  0.000984  0.001231  0.000984  0.001066 | 0.288023
  0.000000  0.171288  0.002379  0.010829  0.000082 | 0.184578
  0.000000  0.000574  0.119196  0.001231  0.009680 | 0.130681
  0.000164  0.007219  0.008121  0.135603  0.000082 | 0.151189
  0.001477  0.000000  0.015833  0.000082  0.228138 | 0.245529
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

分割法の場合

```
Percentage accuracy for tree 1 = 84.3396 +/- 0.329166
Mean square error for tree 1 = 0.253227
Expected accuracy for tree 1 = 84.3396
Leaf count for tree 1 = 96, expected = 96.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.261444  0.002543  0.007301  0.007301  0.007465 | 0.286054
  0.003199  0.161772  0.007055  0.023626  0.002215 | 0.197867
  0.003281  0.002215  0.096801  0.006645  0.015176 | 0.124118
  0.006645  0.013043  0.013536  0.110993  0.001805 | 0.146021
  0.010829  0.000492  0.022067  0.000164  0.212387 | 0.245939
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

VQ コードブック 128 の場合
ベイズ推定の場合

```
Percentage accuracy for tree 1 = 96.0459 +/- 0.176506
Mean square error for tree 1 = 0.0731114
Expected accuracy for tree 1 = 90.852
Typical std. dev. of expected accuracy for an example = 9.80357
Neg. Log Posterior for tree 1 = 4582.67 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 2801, expected = 981.918213
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.284906  0.000984  0.001313  0.000984  0.001231 | 0.289418
  0.000000  0.176456  0.001477  0.009106  0.000082 | 0.187121
  0.000000  0.000492  0.136998  0.002871  0.011403 | 0.151764
  0.000000  0.002133  0.000820  0.135767  0.000000 | 0.138720
  0.000492  0.000000  0.006153  0.000000  0.226333 | 0.232978
-----------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

分割法の場合

```
Percentage accuracy for tree 1 = 87.5554 +/- 0.298972
Mean square error for tree 1 = 0.205094
Expected accuracy for tree 1 = 87.5554
Leaf count for tree 1 = 250, expected = 250.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.267022  0.000328  0.000820  0.001805  0.006399 | 0.276374
  0.002215  0.166694  0.006399  0.018048  0.002379 | 0.195734
  0.001969  0.003199  0.115669  0.007629  0.017063 | 0.145529
  0.007301  0.006645  0.008368  0.116243  0.003281 | 0.141838
  0.006891  0.003199  0.015505  0.005004  0.209926 | 0.240525
-----------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

入力すべての場合

```
Percentage accuracy for tree 1 = 91.0254 +/- 0.258873
Mean square error for tree 1 = 0.146938
Expected accuracy for tree 1 = 91.0254
Leaf count for tree 1 = 531, expected = 531.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.273995  0.001805  0.002297  0.003281  0.006235 | 0.287613
  0.001395  0.169729  0.005332  0.013126  0.002543 | 0.192125
  0.001805  0.002461  0.123216  0.003856  0.011321 | 0.142658
  0.005250  0.005414  0.006727  0.126825  0.002461 | 0.146678
  0.002953  0.000656  0.009188  0.001641  0.216489 | 0.230927
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

前後のラベルを加えた場合
ベイズ推定の場合

```
Percentage accuracy for tree 1 = 98.187 +/- 0.120842
Mean square error for tree 1 = 0.0435909
Expected accuracy for tree 1 = 92.396
Typical std. dev. of expected accuracy for an example =
9.61091
Neg. Log Posterior for tree 1 = 4580.11 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 3374, expected = 1812.810181
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.282937  0.000000  0.000082  0.000000  0.000082 |0.283101
  0.000082  0.177112  0.000000  0.001313  0.000082 |0.178589
  0.000328  0.000082  0.138720  0.000328  0.002625 |0.142084
  0.001477  0.002461  0.003527  0.147006  0.000164 |0.154635
  0.000574  0.000410  0.004430  0.000082  0.236095 |0.241591
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 |1.00000
```

分割法の場合

```
Percentage accuracy for tree 1 = 92.1985 +/- 0.242912
Mean square error for tree 1 = 0.13724
Expected accuracy for tree 1 = 92.1985
Leaf count for tree 1 = 196, expected = 196.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.276538  0.000328  0.001231  0.002625  0.006973 |0.287695
  0.000410  0.168417  0.001395  0.006317  0.000738 |0.177276
  0.001559  0.000246  0.125021  0.003199  0.011403 |0.141427
  0.002379  0.007957  0.006235  0.133962  0.001887 |0.152420
  0.004512  0.003117  0.012879  0.002625  0.218048 |0.241181
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 |1.00000
```

ホルマント情報を3分割した場合の出力

```
<====>
       -----
             -----
```

ベイズ推定

```
Percentage accuracy for tree 1 = 88.9171 +/- 0.284326
Mean square error for tree 1 = 0.166247
Expected accuracy for tree 1 = 86.588
Typical std. dev. of expected accuracy for an example = 7.50975
Neg. Log Posterior for tree 1 = 5352.99 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 2628, expected = 465.966125
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.273585   0.000328   0.000984   0.003363   0.001231 | 0.279491
  0.000000   0.167760   0.002953   0.020263   0.000246 | 0.191222
  0.001313   0.002379   0.113208   0.005660   0.020427 | 0.142986
  0.004348   0.009434   0.010993   0.118704   0.001231 | 0.144709
  0.006153   0.000164   0.018622   0.000738   0.215915 | 0.241591
-------------------------------------------------------------
  0.285398   0.180066   0.146760   0.148728   0.239048 | 1.00000
```

分割法

```
Percentage accuracy for tree 1 = 85.3486 +/- 0.320284
Mean square error for tree 1 = 0.224038
Expected accuracy for tree 1 = 85.3486
Leaf count for tree 1 = 199, expected = 199.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.260623   0.001395   0.004430   0.005824   0.004020 | 0.276292
  0.000246   0.160377   0.003445   0.020180   0.000656 | 0.184906
  0.003035   0.001148   0.107957   0.007711   0.022313 | 0.142166
  0.009024   0.016653   0.010090   0.114520   0.002051 | 0.152338
  0.012469   0.000492   0.020837   0.000492   0.210008 | 0.244299
-------------------------------------------------------------
  0.285398   0.180066   0.146760   0.148728   0.239048 | 1.00000
```

```
-----
     <=====>
              -----
```

ベイズ推定

```
Percentage accuracy for tree 1 = 90.9844 +/- 0.259405
Mean square error for tree 1 = 0.134494
Expected accuracy for tree 1 = 88.8907
Typical std. dev. of expected accuracy for an example = 6.72134
Neg. Log Posterior for tree 1 = 4436.08 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 2620, expected = 379.447968
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.278999  0.000082  0.000492  0.000820  0.000902 | 0.281296
  0.000410  0.170550  0.005660  0.017473  0.000410 | 0.194504
  0.000820  0.002133  0.118212  0.004348  0.021575 | 0.147088
  0.002461  0.007219  0.007629  0.126087  0.000164 | 0.143560
  0.002707  0.000082  0.014766  0.000000  0.215997 | 0.233552
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

分割法

```
Percentage accuracy for tree 1 = 88.4249 +/- 0.289766
Mean square error for tree 1 = 0.177093
Expected accuracy for tree 1 = 88.4249
Leaf count for tree 1 = 164, expected = 164.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.270139  0.001148  0.002625  0.001641  0.001559 | 0.277112
  0.000492  0.169647  0.006809  0.022477  0.000574 | 0.200000
  0.001723  0.001887  0.110336  0.004020  0.022724 | 0.140689
  0.005578  0.007219  0.011321  0.120591  0.000656 | 0.145365
  0.007465  0.000164  0.015669  0.000000  0.213536 | 0.236833
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

```
-----
     ------
            <=====>
```

ベイズ推定

```
Percentage accuracy for tree 1 = 85.3815 +/- 0.319987
Mean square error for tree 1 = 0.227454
Expected accuracy for tree 1 = 77.9782
Typical std. dev. of expected accuracy for an example = 12.525
Neg. Log Posterior for tree 1 = 8754.59 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 4579, expected = 1529.898560
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.263495  0.000574  0.002379  0.004594  0.003199 | 0.274241
  0.000820  0.160705  0.011813  0.022888  0.002297 | 0.198523
  0.006809  0.007957  0.102379  0.009270  0.015340 | 0.141756
  0.004184  0.009926  0.010418  0.110172  0.001148 | 0.135849
  0.010090  0.000902  0.019770  0.001805  0.217063 | 0.249631
-----------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

分割法

```
Percentage accuracy for tree 1 = 72.3954 +/- 0.404897
Mean square error for tree 1 = 0.389839
Expected accuracy for tree 1 = 72.3954
Leaf count for tree 1 = 164, expected = 164.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.230681  0.001887  0.004676  0.009926  0.013700 | 0.260870
  0.007547  0.153322  0.024692  0.040033  0.009598 | 0.235193
  0.011403  0.008696  0.077276  0.016079  0.023462 | 0.136916
  0.015587  0.014192  0.014930  0.079820  0.009434 | 0.133962
  0.020180  0.001969  0.025185  0.002871  0.182855 | 0.233060
-----------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

```
-----
     ------
          ------

<==========>
```

ベイズ推定

```
Percentage accuracy for tree 1 = 93.7408 +/- 0.219393
Mean square error for tree 1 = 0.0980422
Expected accuracy for tree 1 = 90.3815
Typical std. dev. of expected accuracy for an example = 8.07146
Neg. Log Posterior for tree 1 = 4091.81 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 2768, expected = 588.378113
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.283265  0.000082  0.000082  0.000000  0.000328 | 0.283757
  0.000000  0.174487  0.002379  0.011649  0.000164 | 0.188679
  0.000164  0.000738  0.122395  0.000656  0.017145 | 0.141099
  0.001231  0.004676  0.008614  0.136259  0.000410 | 0.151189
  0.000738  0.000082  0.013290  0.000164  0.221001 | 0.235275
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

分割法

```
Percentage accuracy for tree 1 = 88.9828 +/- 0.283588
Mean square error for tree 1 = 0.17436
Expected accuracy for tree 1 = 88.9828
Leaf count for tree 1 = 214, expected = 214.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.269073  0.001313  0.001641  0.001641  0.001395 | 0.275062
  0.000328  0.168581  0.004102  0.018130  0.000656 | 0.191797
  0.002215  0.001395  0.109926  0.003117  0.019032 | 0.135685
  0.006973  0.008368  0.012551  0.125677  0.001395 | 0.154963
  0.006809  0.000410  0.018540  0.000164  0.216571 | 0.242494
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

```
-----
     ------
          ------

     <============>
```

ベイズ推定

```
Percentage accuracy for tree 1 = 94.2576 +/- 0.210719
Mean square error for tree 1 = 0.098804
Expected accuracy for tree 1 = 88.8116
Typical std. dev. of expected accuracy for an example = 10.1402
Neg. Log Posterior for tree 1 = 5041.33 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 3337, expected = 1051.108643
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.283019  0.000000  0.000082  0.000000  0.000082 | 0.283183
  0.000000  0.167760  0.002625  0.007629  0.000082 | 0.178097
  0.000328  0.001477  0.122806  0.003117  0.007629 | 0.135357
  0.000574  0.010829  0.005332  0.137982  0.000246 | 0.154963
  0.001477  0.000000  0.015915  0.000000  0.231009 | 0.248400
-----------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

分割法

```
Percentage accuracy for tree 1 = 84.7908 +/- 0.325256
Mean square error for tree 1 = 0.238222
Expected accuracy for tree 1 = 84.7908
Leaf count for tree 1 = 256, expected = 256.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.264725  0.000082  0.001231  0.002625  0.007055 | 0.275718
  0.003445  0.159393  0.009270  0.021821  0.003199 | 0.197129
  0.003445  0.007219  0.106235  0.008121  0.019114 | 0.144135
  0.007137  0.010008  0.010418  0.111567  0.003692 | 0.142822
  0.006645  0.003363  0.019606  0.004594  0.205989 | 0.240197
-----------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

もとの韻律情報を加えた場合
ベイズ推定の場合

```
Percentage accuracy for tree 1 = 97.3913 +/- 0.144368
Mean square error for tree 1 = 0.0577264
Expected accuracy for tree 1 = 91.552
Typical std. dev. of expected accuracy for an example = 10.1146
Neg. Log Posterior for tree 1 = 4378.9 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 2250, expected = 1043.555908
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.284906  0.000000  0.000000  0.000000  0.000082 | 0.284988
  0.000000  0.177851  0.000984  0.006071  0.000082 | 0.184988
  0.000000  0.000246  0.139705  0.000164  0.009680 | 0.149795
  0.000164  0.001969  0.000492  0.142494  0.000246 | 0.145365
  0.000328  0.000000  0.005578  0.000000  0.228958 | 0.234865
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

分割法の場合

```
Percentage accuracy for tree 1 = 86.0541 +/- 0.313767
Mean square error for tree 1 = 0.232797
Expected accuracy for tree 1 = 86.0541
Leaf count for tree 1 = 192, expected = 192.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.262264  0.004348  0.007055  0.010090  0.007793 | 0.291550
  0.003035  0.164069  0.006809  0.018212  0.002379 | 0.194504
  0.002707  0.002215  0.110418  0.003938  0.018048 | 0.137326
  0.007957  0.008614  0.007711  0.116079  0.003117 | 0.143478
  0.009434  0.000820  0.014766  0.000410  0.207711 | 0.233142
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

分類された韻律情報を加えた場合
ベイズ推定の場合

```
Percentage accuracy for tree 1 = 96.6612 +/- 0.162712
Mean square error for tree 1 = 0.0659579
Expected accuracy for tree 1 = 91.3976
Typical std. dev. of expected accuracy for an example = 9.93047
Neg. Log Posterior for tree 1 = 4370.62 (nits)
Neg. Log Posterior for examples = 19203.9 (nits)
Leaf count for tree 1 = 2540, expected = 987.155396
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.283921  0.000574  0.000738  0.000820  0.000164 | 0.286218
  0.000000  0.175636  0.000984  0.004430  0.000246 | 0.181296
  0.000820  0.000328  0.136998  0.000820  0.011239 | 0.150205
  0.000164  0.003527  0.000902  0.142658  0.000000 | 0.147252
  0.000492  0.000000  0.007137  0.000000  0.227400 | 0.235029
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

分割法の場合

```
Percentage accuracy for tree 1 = 84.8072 +/- 0.325112
Mean square error for tree 1 = 0.246677
Expected accuracy for tree 1 = 84.8072
Leaf count for tree 1 = 156, expected = 156.000000
Misclassification matrix for tree 1:
 (row = predicted, col = actual, in .attr file order)
  0.262510  0.003938  0.005742  0.008696  0.012551 | 0.293437
  0.004184  0.162920  0.007547  0.023790  0.003117 | 0.201559
  0.002543  0.002789  0.109516  0.005086  0.017801 | 0.137736
  0.008942  0.009762  0.009434  0.110254  0.002707 | 0.141099
  0.007219  0.000656  0.014520  0.000902  0.202871 | 0.226169
-------------------------------------------------------------
  0.285398  0.180066  0.146760  0.148728  0.239048 | 1.00000
```

付録 C

データベース削減のための、処理の流れ

Sun 用 503 文 Wave データの作成 (付録 D D-1 参照)

⇓

Wave データより、ホルマント情報を求める。(付録 A D-1 参照)

⇓

ラベルファイルより、各母音に対するホルマント情報の切り出しを行なう。(D-2 参照)

⇓

切り出された、ホルマント情報を 1 つのファイルにする。
(copysps によって行なう。)

⇓

1 つにまとめられたファイルに対して、VQ コードブックの作成を行なう。(付録 A D-2 参照)

⇓

作成された VQ コードブックより VQ を行なう。(付録 A D-3 参照)

⇓

VQ によって求められた、VQ コードベクトル番号に対して再度ラベルファイルより、各母音に対する情報に直す。(D-4 参照)

⇓

各母音の情報に直された、VQ コードベクトル番号に対してヒストグラムを求める。
(D-5 参照)

⇓

求められたヒストグラムを用いて、Tree の作成を行なう。(付録 A D-6 参照)

⇓

作成された Tree に対して、Leaf の出力結果の参照を行なう。

付録 D (Shell & Program)

- D-1 -

Sun ワークステーション用 Wavefile コンバート
($DEC \leftrightarrow Sun(CSHELL)$)
(503 文データの場合)
(**** 注意 *** 最後の J の 3 文が取り込まれない (書き換え必要))

```
# !/bin/csh
 set ESPS=/usr/local/ESPS/bin/
 set DB_LBL=/DB/Bset1/MTK/LBL/SD/
 set DB_WAV=/DB/Bset1/MTK/WAV/SD/
 set HOME=~/MTK/    (* 各自のホームディレクトリ *)
   (* 今は MTK のデータ *)

foreach i(A B C D E F G H I J)
@ count=1

while ($count <= 50)
  echo $count
  if ($count < 10) then
    echo "under 0"
    set num1 = $DB_WAV$i/"MTK_SD_"$i"0"$count
    set num2 = $DB_LBL$i/"MTK_SD_"$i"0"$count
    set num3 = $HOME$i/"MTK_SD_"$i"0"$count
  else
    set num1 = $DB_WAV$i/"MTK_SD_"$i$count
    set num2 = $DB_LBL$i/"MTK_SD_"$i$count
    set num3 = $HOME$i/"MTK_SD_"$i$count
  endif

  set  fn = $num1".12K"
  echo $fn

  set fn2 = $num3".d"
  echo $fn2

  set fn3 = $num2".LB"
```

```
    echo $fn3

    dd if=$fn conv=swab > $num3.tmp
    echo "OK"
    $ESPS/btosps -f 12000 -c " " $num3.tmp  $fn2
    echo "OK"
    $ESPS/formant $fn2
    echo "OK"
    rm $num3.tmp
    @ count++
    end
end
```

- D-2 -

ホルマント切り出し (CSHELL & awk)

(これは ESPS で出力された fb ファイルに対する母音の切り出し用の Shell & awk です。)

```
-----CSHELL------
# ! /bin/csh
 set ESPS=/usr/local/ESPS/bin/
 set DB_LBL=/DB/Bset1/MTK/LBL/SD/
 set DB_WAV=/DB/Bset1/MTK/WAV/SD/
 set HOME1=~/MTK/
 set HOME2=~/for2/  (* 入れたい Directory へ *)

foreach i(A B C D E F G H I J)
@ count=1
@ kazu=1

while ($count <= 50)
  echo $count
  if ($count < 10) then
    echo "under 0"
    set num1 = $DB_WAV$i/"MTK_SD_"$i"0"$count
    set num2 = $DB_LBL$i/"MTK_SD_"$i"0"$count
    set num3 = $HOME1$i/"MTK_SD_"$i"0"$count
    set num4 = $HOME2$i/"MTK_SD_"$i"0"$count
  else
    set num1 = $DB_WAV$i/"MTK_SD_"$i$count
    set num2 = $DB_LBL$i/"MTK_SD_"$i$count
    set num3 = $HOME1$i/"MTK_SD_"$i$count
    set num4 = $HOME2$i/"MTK_SD_"$i$count
  endif
  set  fn = $num1".12K"
  echo $fn

  set fn2 = $num3".d"
  echo $fn2

  set fn3 = $num2".LB"
  echo $fn3
```

```
foreach j(a i u e o)
  rm $num4/$j.*
  set fn4 = $num4".fb_"$j
  set fn5 = $num3".fb"

  nawk -f cut.awk $fn3 > tmp1    (* ここでの awk は次に示す *)
  echo "nawk1 OK"
  cp tmp1 tmp5
  grep " $j" tmp5  > tmp2
  echo $j

  echo $fn5

  cp $fn5 tmp.fb

  cat -b tmp2 > tmp3
  nawk -f cut3.awk tmp3  (* 同様に後で示す *)
  mkdir $num4
  mv $j*  $num4

  set fn7 = $num4"/MTK_SD_"$i$count".LB2" (* 2層目のラベルファイルも切り出
す *)
  mv tmp1 $fn7

  echo "nawk2 OK"
  end
 @ count ++
 end
end

-----awk files----
{* cut.awk *}

BEGIN{ while ($1 != "#") getline;}
{
       if (NF == 1)
  exit
          ph = $2;
          st = $1;
```

```
          ed = $3 -$1;
  ed2 = $3;
         print int(st), int(ed2), ph
}
{* cut3.awk *}

{
          ph = $4;
          st = $2;
          ed2 = $3;
          n = $1;

        cmd = "/usr/local/ESPS/bin/fea_deriv -r " int(st/10)":" int(ed2/10) \
              " fields tmp.fb out.fb"   (* fields ファイルが必要 *)
        cmd2 = "mv out.fb " $4"."n
  print cmd
  print cmd2

  system(cmd)
  system(cmd2)
}

{* fields ファイル *}
field=a1
fm[0:3]
pfield=b1
bw[0:3]
(* field 名は何でも良い この場合は第 1 〜第 4 ホルマントの 4vector x 2 *)
```

- D-3 -

VQ のための Shell

```
# ! /bin/csh
 set ESPS=/usr/local/ESPS/bin/
 set HOME1=~/for1/
 set HOME2=~/Waves/


foreach i(v)

        $ESPS/fea_deriv ~/Waves/fields2 big.v.fob fb.v2
        $ESPS/vqdes -x 3 -P ~/vqparms fb.v2 $i.vq2
        $ESPS/vq -f fb_f0  $i.vq2 fb.v2 fb_out2.$i
        $ESPS/psps fb_out2.$i | grep cwndx | nawk '{print $2}' > $i.vq3
        cat -n $i.vq3 > ~/Waves/out.$i.256
end


{*  fields2 *}
field=fb
ptype=FLOAT
a1[0:3]
b1[0:3]

{* vparams *}
# @(#)Pvqdes    3.1 10/8/87 ESI
# default parameter file for vqdes
# string fea_field      = ''LPCEPSTRA'';
  string fea_field      = ''fb'';
int fea_dim             = 8;   (* 符号化したいベクトルが何次元であるか *)
float conv_ratio        = .05;
int vq_size             = 256;    (* VQ codebook size *)
int max_iter            = 100;
string dist_type        = ''MSE'';
string cbk_struct       = ''FULL_SEARCH'';
string cbk_type         = ''MISC'';
string init_file        = ''vqdes.cbk'';
int init_rec            = 1;
string init_behav       = ''INIT_CLUSTER'';
```

- D-4 -
各母音情報における、VQ コードベクトル番号への変換
(これは、一つの例であって、そのままでなくてよい
またこれは、/o/ の場合のみであり、各母音に対して行なうこと)

```
# ! /bin/csh

 @ kazu1 = 1
 @ kazu2 = 0
 @ counts = 1
 @ loop = 1
 @ loop3 = 1
 set HOME2=~/for2/

foreach i(A B C D E F G H I J)
@ count=1

   while ($count <= 50)
      if ($count < 10) then
          set num4 = $HOME2$i/''MTK_SD_''$i''0''$count
          echo $num4
      else
         set num4 = $HOME2$i/''MTK_SD_''$i$count
         echo $num4
     endif
     @ loop3 = 1
     while (-e $num4/o_$loop3)
         @ loop = $kazu1

         set x = `/usr/local/ESPS/bin/psps $num4/o_$loop3
| grep ``Number''| awk '{print $5}''
         echo $x
         @ kazu2 = $x + $kazu2
         echo $num4''/o_''$loop3 >> data_o4
         while ($loop <= $kazu2 )
            set z  = `sed -ne ''$loop'p' ~/Waves/out.o.32
| awk '{print $2}''
            echo $x'' 1 ``$z
            echo $x'' 1 ``$z >> data_o4
```

```
            @ loop ++
        end
        @ kazu1 = $kazu1 + $x
        @ counts ++
        @ loop3 ++
    end
@ count ++
end
end
```

- D-5 -

ヒストグラムの作成

これは、D-3 によって求められた各母音における情報の書式に基づいて書いている。

```
BEGIN{i=1;vq2=$1 getline;}
{
        if(NF>1)
        {
                vq[i]=$3
                i++
                d = i
                suu=$1
        }
        if(NF==1)
        {
                for(i=1;i<=d-1;i++){ sum+=vq[i]}
                print sum/suu
                sum = 0
                i=1
        }
}
END{
        for(i=1;i<=d-1;i++){ sum+=vq[i] }
        s=sum/suu
        print s
}
```

- D-6 -

・Tree 入力ファイルの作成

これは、D-4 によって求められた各母音における情報の書式に基づいて書いている。

```
BEGIN{vq=0}
{
        vq =vq ‘‘ ‘‘$1;
}
END{print vq}
```

これにより、Tree 入力時の書式になる。

実際入力に対して、本来の出力を書くので次に示すような書式となる。(よって、本来の出力を Shell の出力に付け足すこと)

例 VQ codebook 8 (formant only)

```
a 0 0 2 0 2 0 0 6
a 4 0 0 0 5 1 0 0
i 0 0 0 0 0 0 0 10
        .
        .
        .
```

例 codebook 8 (formant + f0 + dulation)

```
  f0 dl     hist
a  1 1  0 0 2 0 2 0 0 6
a  2 1  4 0 0 0 5 1 0 0
i  3 1  0 0 0 0 0 0 0 10
        .
        .
        .
```

・Tree 作成

入力データすべて

nick/bin/mktree -v -s c4 -c “-slt” ファイル名

分割法 (3 分割)

nick/bin/mktree -v -o “-C 3” -s c4 -c “-slt” ファイル名

ベイズ推定

nick/bin/mktree -v -c “-slt” -s bayes ファイル名