TR-I-0375

HMM-LR における各種探索手法ユーザーズマニュアル

User's Manual for the HMM-LR Implemented Using Various Search Techniques

山口 耕市
Kouichi YAMAGUCHI

1993 年 3 月

概 要

従来、HMM-LR 連続音声認識システムでは固定的なビーム探索が用いられていた。本レポートでは、適応的ビーム探索および A^* アルゴリズムに基づく探索など高速化をめざした探索手法を用いたときの混合連続分布型 HMM-LR 連続音声認識システムの使用方法を説明する。また、Cシェルスクリプトで書かれた実際の使用例も添付する。

目次

第 1 章

はじめに

混合連続分布型 HMM-LR の学習・認識を行なうプログラムの使用方法について説明する。混合連続分布型 HMM-LR は離分布散型 HMM-LR に比べ、計算量は多いものの高精度な音素モデルを構築することができる [1, 6, 8]。また、話者適応や不特定話者への発展性も期待されている。

HMM-LR 連続音声認識システムでは標準的には、ビームサーチが用いられている [2, 1]。ところが、一般的にビームサーチは最適性が保証されていない上、高い認識率を得るためにはビーム幅を大きくしなければならなく、多くの音素照合回数を必要とする。特に、継続時間長制御をしない場合、その傾向が強い [5]。現在の音声認識システムでは音素モデルの性能が向上し正解候補の順位は 10 位以内に入っていることが多い。しかし、音素認識が完璧になされるわけではないので、正解候補の順位はときどき 100 位を超えることもある。したがって高い認識性能を得るためには大きなビーム幅が必要となる。結果として無駄な探索を行なっていることが多いと言える。

そこで北は可変ビームサーチ [3] を、山口等は A^* アルゴリズムに基づく探索 [6, 7] や best-first 探索法、学習可能な適応的ビームサーチ [8, 9] を導入した。

このソフトウェアパッケージはCのプログラムとCシェルスクリプトからできている。この章の残りでは高速サーチ版 CMHMM-LR ソフトウェアパッケージの使用上の制限について述べる。第 2 章では学習可能な適応的ビームサーチを用いた場合、第 3 章では A^* アルゴリズムに基づく探索を用いた場合の HMM-LR 認識プログラムの使い方についてそれぞれ述べるとともに、文節音声認識実験の実行例も示す。

なお、このソフトウェアパッケージで扱う HMM-LR 連続音声認識システムは、混合連続分布型で無相関正規分布を仮定した対角成分のもののみを対象としている。HMM-LR の全般的な取り扱い説明については、文献 [4] を参照してください。混合連続分布型 HMM-LR については、文献 [10] を参照してください。

1.1 使用上の注意

このソフトウェアは DECstation 5000/25(OS は ULTRIX V4.2A) 上でコンパイルされ、動作が確認されている。なお、適応的ビームサーチで使用するニューラルネットワークの学習および実行は HP9000/730 上でも動作が確認されている。同サーチで使用する重回帰分析は、ソフトウェアパッケージ "Numerical Recipes in C: The Art of Scientific Computing" (Cambridge University Press, 1988) を用いている。必要とされるメインメモリはタスクに依存する。動作を確認した我々のワークステーションは 64MB のメモリを実装している。添付してあるデモソフトを実行するには、`nawk`, `sed`, `csh` などの UNIX プログラムが必要である。また、HMM-LR 用実験プログラム、ツール類はディレクトリ `CMHMM-LR` にある。

第 2 章

適応的ビームサーチ

2.1 原理

一般に、正解仮説を含む最小の探索空間 $\Theta(t)$ が得られれば探索の効率は最も良くなる。これはビーム探索の場合、正解仮説の順位が分かっており、ビーム幅を正解仮説の順位に設定することに相当する。そこで観測可能な特徴量 O_i を変数とする m 変数関数 $\varphi(\,)$ を使って、この最小の探索空間 $\Theta(t)$ を式 (2.1) のように近似する。

$$\widehat{\Theta}(t) = \varphi(O_1(t), O_2(t), \cdots, O_m(t)), \tag{2.1}$$

ここで、$\{O_i(t)\}(i = 1, 2, \cdots, m)$ は時刻 t における観測可能な特徴量の集合。なお、従来のビーム探索の場合、式 (2.2) のように $\widehat{\Theta}(t)$ は定数関数と同じになる。

$$\widehat{\Theta}(t) \equiv const. \tag{2.2}$$

従来から用いられてきたヒューリスティック探索では、主として仮説のスコアのみをヒューリスティック情報として用いてきた。それに対し、本発明では従来のヒューリスティック情報に加え、各仮説のスコアの分布状況等の観測可能な特徴量を入力とする制御関数 $\varphi(\,)$ を基に枝刈りすべきか否かを決定する。すなわち探索範囲を適応的に変化させる。この制御関数 $\varphi(\,)$ はニューラルネットワークまたは重回帰分析で構成されており、認識実験で得られる観測可能な特徴量と正解仮説の順位のサンプルを使って予め学習しておく。

予備実験の結果、正解仮説の順位と観測可能な特徴量との間には何らかの相関関係が認められており、制御関数 $\varphi(\,)$ は精度良く正解仮説の順位を予測できることが期待できる。

2.2 HMM-LR への適用

HMM-LR システムにおいて、ビーム探索は音素同期で動作する。すなわち解析木の深さが 1 つ進む度に枝刈りを実行する。したがって式 (2.1) において、時刻 t は解析木の深さ n で代用される。

ニューラルネットワークはときどき真の順位よりかなり大きい値を出力することがある。このような過大評価は逆に無駄な探索につながる。そこでニューラルネットワークよりはきめ細かいビーム幅の制御はできなくても、過大評価に陥る危険が少ない、つまりロバスト性のある制御関数を用いて上限値を設定することが考えられる。すなわち、複数種類の制御関数の組み合わせでビーム幅を決定する。この実施例では北によって導入された可変ビーム探索 [3] と組み合わせる。式 (2.3) のように各深さ n でニューラルネットワークの出力と可変ビーム探索による出力を比較し、値の小さい方を選択する。ただし、過小評価対策として小さな値 $margin$ を用いている。

$$\Phi(n) \;=\; \min(\varphi_V(n),\, \varphi_N(O_1(n), O_2(n), \cdots, O_m(n)) + margin), \tag{2.3}$$

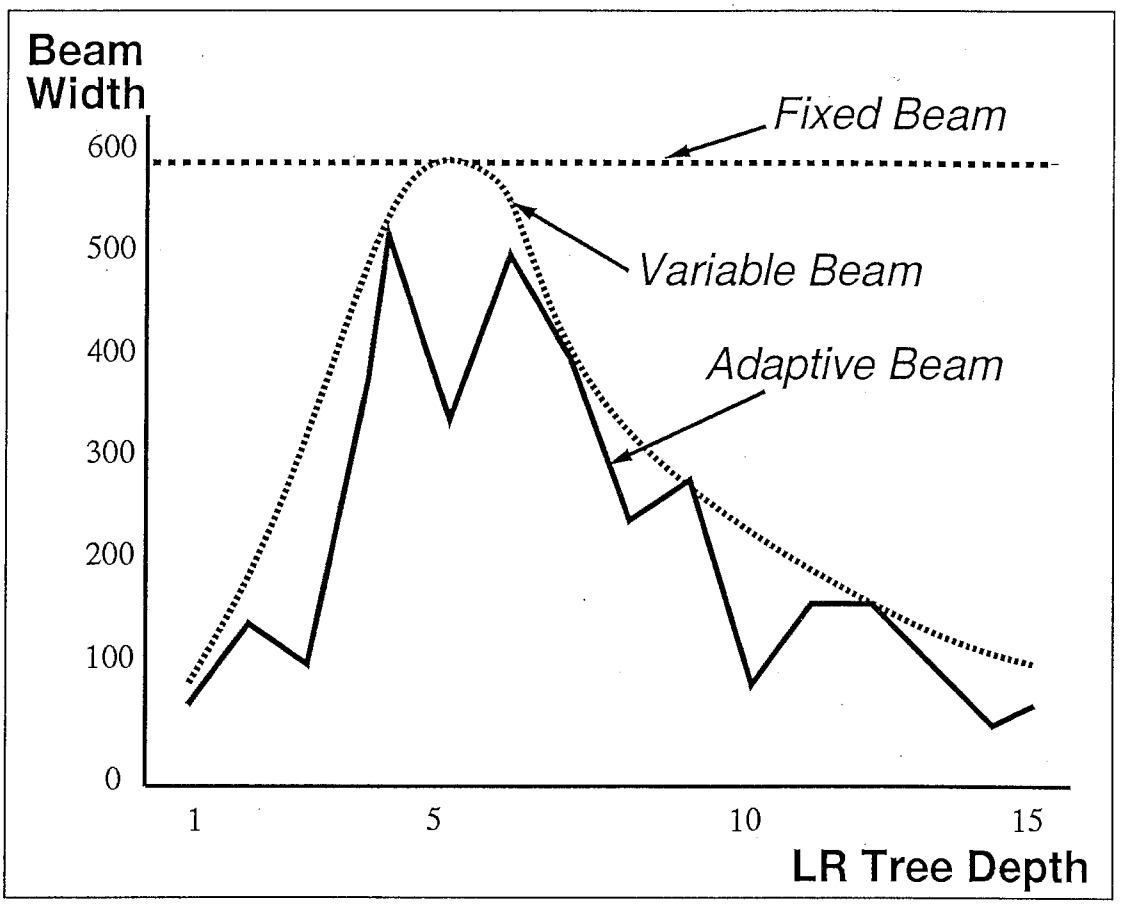

図 2.1: 3つの探索手法の比較

ここで、$\varphi_V()$ は可変ビーム探索、$\varphi_N()$ はニューラルネットワーク、$\Phi(n)$ は本発明による適応的ビーム探索である。この組み合わせ方式を図 2.1に示す。図中、可変ビーム探索は従来の固定ビーム探索に比べ山の両側部分の探索空間を、適応的ビーム探索はさらに細かく探索空間を削減できる。

2.3 学習サンプルの作成

適応的ビームサーチを HMM-LR に適用した場合のシステム構成を図 2.2に示す。
仮説のスコアの分布は次式 (2.4),(2.5) のように回帰分析によってなされる。

$$y = a_1 x + a_0, \tag{2.4}$$

$$y = b_2 x^2 + b_1 x + b_0, \tag{2.5}$$

ここで、x は仮説の順位、y はスコアの近似値。便宜上、x は $\{1, 2, \cdots, 10\}$ に制限されている。

以下の使用例は観測可能な特徴量、すなわち制御関数 $\varphi()$ への入力 $\{O_1, O_2, \cdots, O_m\}$ として次の9つの観測量を抽出する。

- 回帰係数: $a_0, a_1, b_2, b_1, \Delta a_1$
- LR 解析木の現在の深さ: n
- 現在の探索時点 (深さ n) の第1位仮説のスコアと直前の探索時点 (深さ $n-1$) の第1位仮説のスコアとの差: $\Delta score$

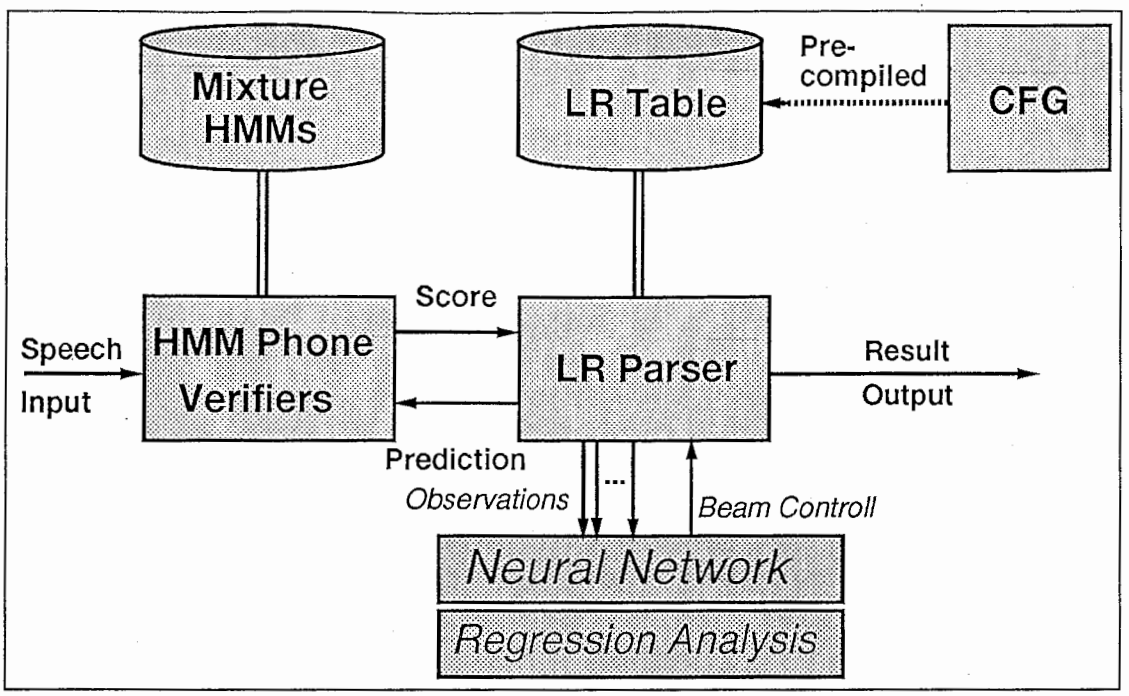

図 2.2: HMM-LR 連続音声認識システム構成

- 第 1 位仮説のスコアと正解仮説のスコアとの差: 1-S （参考データ）
- 1 モーラあたりの平均フレーム数: mora （参考データ）
- 枝分かれ数: branch （参考データ）

2.3.1 観測量の抽出

まず HMM-LR の文節音声認識実験を通して観測可能な特徴量を抽出する。作成にあたっては、プログラム CMHMM-LR/Src/observe/main を用いる。サンプルの作成なので、ビーム幅は大きめに設定する。下の例では 900 にしてある。観測量は抽出結果の例中、HMM の尤度計算時間の表示の下に印字されている。左から順に、正解あり (C) ／なし (E)、木の深さ、正解候補の順位、正解音素列、第 1 位仮説のスコアと正解仮説のスコアとの差、$\Delta score$、a_1、Δa_1、1 モーラあたりの平均フレーム数、b_2、b_1、a_0、枝分かれ数である。

【使用例】

```
$ cd CMHMM-LR/Exe
$ ../Src/observe/main \
  ../DATA/MSH_List_SB3 \
  -g ../GRAMMAR/sp2 \
  -m ../HMM/MSH/model_list \
  -D ../DATA/para_list.MSH \
  -P 5 -M 1 -B 900 -b 64 -c 3000 \
  -s 2 -S 3 -W 0 -A 7 -v 1.2 -f 0 -p 0 \
  | tee features.MSH
```

【抽出結果の例】

```
::::::::::::::::::::::::
CMHMM-LR/Exe/features.MSH
::::::::::::::::::::::::
(001)  147120 182   |daiikkai|
Making likelihood map: CPU-time = 18436 msec, Elapsed-time = 18 sec.
 dep rank       1-S    S'     a1     a1'   mora   b2     b1     a0    branch
C  1   1 d      0.000 48.177 -6.013 -6.013  6.00 -0.745  1.310 48.089 34
C  2   1 da     0.000 -2.023 -3.503  2.510 27.00 -0.741  3.550 45.986 183
C  3   1 dai    0.000 -0.559 -1.976  1.527 26.00 -0.531  3.031 45.359 1158
C  4   4 daii   1.350  1.114 -1.441  0.535 17.33 -0.486  3.105 46.273 1914
C  5   1 daiiQ  0.000  0.003 -1.366  0.074 24.50 -0.323  1.695 46.438 1640
C  6   1 iiQ-k  0.000 -0.208 -1.157  0.209 24.75 -0.344  2.072 46.216 1783
C  7   1 iQ-ka  0.000 -0.506 -0.769  0.388 24.60 -0.289  1.930 45.740 1421
C  8   1 Q-kai  0.000  0.507 -1.326 -0.557 26.83 -0.491  3.258 45.906 1190
E  9   1 kai-t -0.233 -0.233 -0.221  1.105 28.50 -0.016 -0.052 46.290 1208
************************************************************
Parsing time: CPU-time = 50223 msec, Elapsed-time = 51 sec.
Total-verify = 13375  Depth = 9
  1: daiiQ-kai                (prob = -46.32508)
  2: daitai                   (prob = -45.76401)
  3: nai-kurai                (prob = -45.33813)
  4: da-shi-ta-i              (prob = -45.02445)
  5: kai-kurai                (prob = -44.98305)
************************************************************

(002)  147382 170   |tsuuyaku|
Making likelihood map: CPU-time = 17143 msec, Elapsed-time = 18 sec.
 dep rank       1-S    S'     a1     a1'   mora   b2     b1     a0    branch
C  1   1 ts     0.000 50.825 -5.773 -5.773 10.00 -0.477 -0.912 51.436 34
C  2   1 tsu    0.000 -0.396 -3.182  2.591 11.00 -0.658  3.090 50.269 183
C  3  33 tsuuy  2.062 -1.315 -1.041  2.142 15.00 -0.179  0.680 49.161 1158
C  4  50 suuya  3.315 -0.335 -1.977 -0.936 11.00 -0.053 -1.314 49.208 1695
C  5 123 uuyak  2.904 -1.016 -1.080  0.897 34.67 -0.329  2.011 47.641 1824
C  6   3 uyaku  0.836 -1.295 -0.845  0.235 38.00 -0.241  1.427 46.074 1404
E  7   1 -ru-r  0.236 -0.600 -0.722  0.123 12.33 -0.097  0.221 45.946 1363
************************************************************
Parsing time: CPU-time = 34169 msec, Elapsed-time = 34 sec.
Total-verify = 9784  Depth = 7
  1: tsuuyaku                 (prob = -45.65362)
  2: tsuka-u                  (prob = -43.28194)
  3: tsudu-ku                 (prob = -41.66172)
  4: tsu-ku                   (prob = -41.64082)
  5: tsuku-ru                 (prob = -41.37703)
************************************************************

(003)  147674 123   |deNwa|
Making likelihood map: CPU-time = 12409 msec, Elapsed-time = 13 sec.
 dep rank       1-S    S'     a1     a1'   mora   b2     b1     a0    branch
C  1   1 d      0.000 48.250 -4.870 -4.870  6.00 -0.425 -0.565 48.645 34
C  2   1 de     0.000 -0.007 -2.719  2.151 26.00  0.006 -2.528 48.644 183
C  3   1 deN    0.000  0.242 -1.726  0.993 35.00  0.094 -2.424 48.802 1158
C  4   1 deNw   0.000 -0.189 -0.469  1.257 38.50 -0.141  0.854 48.096 1881
C  5   1 deNwa  0.000 -0.345 -0.734 -0.265 27.00 -0.212  1.264 47.769 1527
E  6   1 Nwa-a -0.220 -0.220 -0.671  0.063 26.25 -0.094  0.248 47.755 1565
```

```
************************************************************
Parsing time: CPU-time = 21494 msec, Elapsed-time = 21 sec.
Total-vẽrify = 8095  Depth = 6
  1: deNwa                      (prob = -47.90800)
  2: zeN-wa                     (prob = -47.62052)
  3: neN-wa                     (prob = -47.54892)
  4: meN-wa                     (prob = -46.87156)
  5: zeN-da                     (prob = -46.86906)
************************************************************

(004)  148002 160   |kokusai|
Making likelihood map: CPU-time = 16143 msec, Elapsed-time = 16 sec.
 dep rank       1-S    S'     a1     a1'   mora   b2     b1     a0     branch
C  1   1 k      0.000 43.930 -3.814 -3.814  7.00 -1.410  9.354 42.150 34
C  2   1 ko     0.000 -0.439 -4.683 -0.868 15.00 -0.554  0.775 44.279 183
C  3   4 kok    1.630 -0.322 -2.479  2.203 30.00 -0.652  3.670 43.283 1158
C  4  13 koku   0.565 -1.269 -0.335  2.144 15.00 -0.143  0.998 41.721 1742
C  5   1 kokus  0.000  1.077 -1.121 -0.785 37.50 -0.436  2.948 42.524 1894
C  6   1 okusa  0.000  0.425 -1.057  0.064 33.33 -0.341  2.142 42.999 1527
C  7   1 kusai  0.000  1.222 -1.894 -0.837 33.75 -0.769  5.272 43.612 1456
E  8   1 sai-t -0.177 -0.177 -0.336  1.558 37.25 -0.094  0.549 44.404 1377
************************************************************
Parsing time: CPU-time = 38548 msec, Elapsed-time = 38 sec.
Total-verify = 11677  Depth = 8
  1: kokusai                    (prob = -44.51477)
  2: kono-sai                   (prob = -43.03665)
  3: koNkai                     (prob = -42.15459)
  4: ano-sai                    (prob = -42.10729)
  5: kokunai                    (prob = -42.00080)
************************************************************

  …… (中略) ……

(278)  254496 150   |yoroshiku|
Making likelihood map: CPU-time = 0 msec, Elapsed-time = 15 sec.
 dep rank       1-S    S'     a1     a1'   mora   b2     b1     a0     branch
C  1   3 y      0.383 42.044 -2.500 -2.500 14.00 -0.022 -2.070 42.502 34
C  2   5 yo     0.129 -0.338 -1.047  1.453 40.00  0.074 -1.627 41.981 183
C  3  21 yo-r   1.231  0.506 -1.142 -0.095 26.00 -0.324  1.907 42.246 1158
C  4  11 yoro   0.878 -0.113 -1.145 -0.003 20.00 -0.267  1.389 42.181 1703
C  5   1 orosh  0.000 -0.201 -0.340  0.804 32.50  0.008 -0.385 41.942 1905
C  6   5 roshi  1.183  1.078 -1.247 -0.907 25.33 -0.423  2.714 42.656 1380
C  7   7 shi-k  1.053 -0.344 -0.923  0.324 29.67 -0.288  1.782 42.298 1363
C  8   1 hi-ku  0.000  0.220 -1.721 -0.798 33.50 -0.263  0.824 43.088 1344
E  9   1 -ku-t -0.175 -0.175 -0.618  1.104 34.00 -0.118  0.510 42.595 1307
************************************************************
Parsing time: CPU-time = 0 msec, Elapsed-time = 40 sec.
Total-verify = 13049  Depth = 9
  1: yoroshiku                  (prob = -42.94055)
  2: doushitsu                  (prob = -41.77451)
  3: cyousyoku                  (prob = -41.77391)
  4: moushiko-mu                (prob = -41.43063)
  5: moshikuha                  (prob = -41.37233)
************************************************************
```

```
(279)  254887 150   |onegai|
Making likelihood map: CPU-time = 0 msec, Elapsed-time = 15 sec.
 dep rank       1-S    S'     a1     a1'   mora   b2     b1     a0    branch
C  1   2 o      0.287 36.162 -3.139 -3.139 10.00 -0.585  2.460 36.372 34
C  2   1 on     0.000  2.622 -2.542  0.597 27.00 -0.338  0.762 38.893 183
C  3   1 one    0.000  3.353 -2.475  0.067 24.50 -0.596  3.165 41.609 1158
C  4   1 oneg   0.000  1.727 -1.822  0.653 32.50 -0.643  4.189 42.761 1979
C  5   3 onega  0.068  0.376 -2.887 -1.064 29.00 -0.435  1.329 44.676 1737
C  6   1 ega-i  0.000  1.803 -1.833  1.053 32.25 -0.650  4.244 45.750 1528
E  7   1 a-i-t -0.007 -0.007 -0.506  1.327 34.75 -0.020 -0.279 46.138 1349
************************************************************
Parsing time: CPU-time = 0 msec, Elapsed-time = 32 sec.
Total-verify = 9808  Depth = 7
  1: onegai                    (prob = -46.14469)
  2: o-re-na-i                 (prob = -44.80836)
  3: negai                     (prob = -44.46238)
  4: o-ke-na-i                 (prob = -44.20879)
  5: o-re-ta-i                 (prob = -44.18129)
************************************************************

(280)  255293 132   |moushi|
Making likelihood map: CPU-time = 0 msec, Elapsed-time = 13 sec.
 dep rank       1-S    S'     a1     a1'   mora   b2     b1     a0    branch
C  1   5 m      1.620 45.626 -5.499 -5.499 13.00 -0.350 -1.817 46.656 34
C  2   2 mo     0.079  0.951 -1.783  3.716 33.00 -0.296  1.068 46.740 183
C  3   2 ou-sh  0.001 -0.518 -1.007  0.776 40.00 -0.137  0.329 46.205 1158
C  4   2 u-shi  0.001 -0.001 -0.872  0.135 33.33 -0.182  0.863 46.154 1802
E  5   1 oshim -0.472 -0.473 -0.179  0.693 50.50 -0.012 -0.050 45.616 1907
************************************************************
Parsing time: CPU-time = 0 msec, Elapsed-time = 18 sec.
Total-verify = 6097  Depth = 5
  1: moshi                     (prob = -45.47733)
  2: moushi                    (prob = -45.47653)
  3: youshi                    (prob = -44.70536)
  4: oki-i                     (prob = -42.37618)
  5: oso-i                     (prob = -42.20249)
************************************************************

(281)  255700 125   |agemasu|
Making likelihood map: CPU-time = 0 msec, Elapsed-time = 13 sec.
 dep rank       1-S    S'     a1     a1'   mora   b2     b1     a0    branch
C  1   1 a      0.000 41.487 -5.385 -5.385  8.00 -1.291  6.839 39.788 34
C  2   1 ag     0.000 -1.042 -1.409  3.977 23.00 -0.298  1.425 40.338 183
C  3   1 age    0.000  1.026 -2.042 -0.633 18.50 -0.625  3.824 41.126 1158
C  4   1 age-m  0.000  0.294 -1.263  0.779 24.50 -0.252  1.138 41.545 1852
C  5   1 ge-ma  0.000 -0.982 -0.708  0.555 18.33 -0.171  0.915 40.732 1642
C  6   1 -ma-s  0.000  0.085 -0.845 -0.138 35.00 -0.183  0.895 40.745 1456
C  7   1 ma-su  0.000 -0.072 -0.844  0.001 26.25 -0.288  1.850 40.527 1289
E  8   1 -su-t  0.201  0.201 -0.978 -0.133 28.50 -0.050 -0.433 41.161 1371
************************************************************
Parsing time: CPU-time = 0 msec, Elapsed-time = 31 sec.
Total-verify = 11571  Depth = 8
  1: age-ma-su                 (prob = -41.06865)
```

```
  2: a-re-ma-su                (prob = -40.73096)
  3: age-na-sou                (prob = -39.76161)
  4: a-ri-ma-su                (prob = -39.54932)
  5: a-re-na-sou               (prob = -39.42393)
************************************************************
```

なお、この観測量の抽出ルーチンで使用した話者はクローズドとなるので、実際の認識実験ではそれ以外の話者を用いることが望ましい。

2.4 ニューラルネットワーク

ニューラルネットワークの構築・学習・評価はディレクトリ CMHMM-LR/NN で行なう。ニューラルネットワークの役割を図 2.3に示す。重回帰分析の役割もこれと同じである。両者の違いは、ニューラルネットワークは非線形システムであるのに対し、重回帰分析は線形システムであることである。

2.4.1 観測量のフォーマット化

抽出結果ファイルをニューラルネットワークで学習するために、フォーマット化された学習用サンプルファイルに変換する。作成にあたっては、ディレクトリ CMHMM-LR/sampling の下で以下のツールを使う。このフォーマットはニューラルネットワーク学習プログラム “DCP2”[11] に準じている。

`mksample.awk` 学習サンプル抽出 AWK プログラム
`mksample.csh` 学習サンプル抽出シェルスクリプト

【使用例】

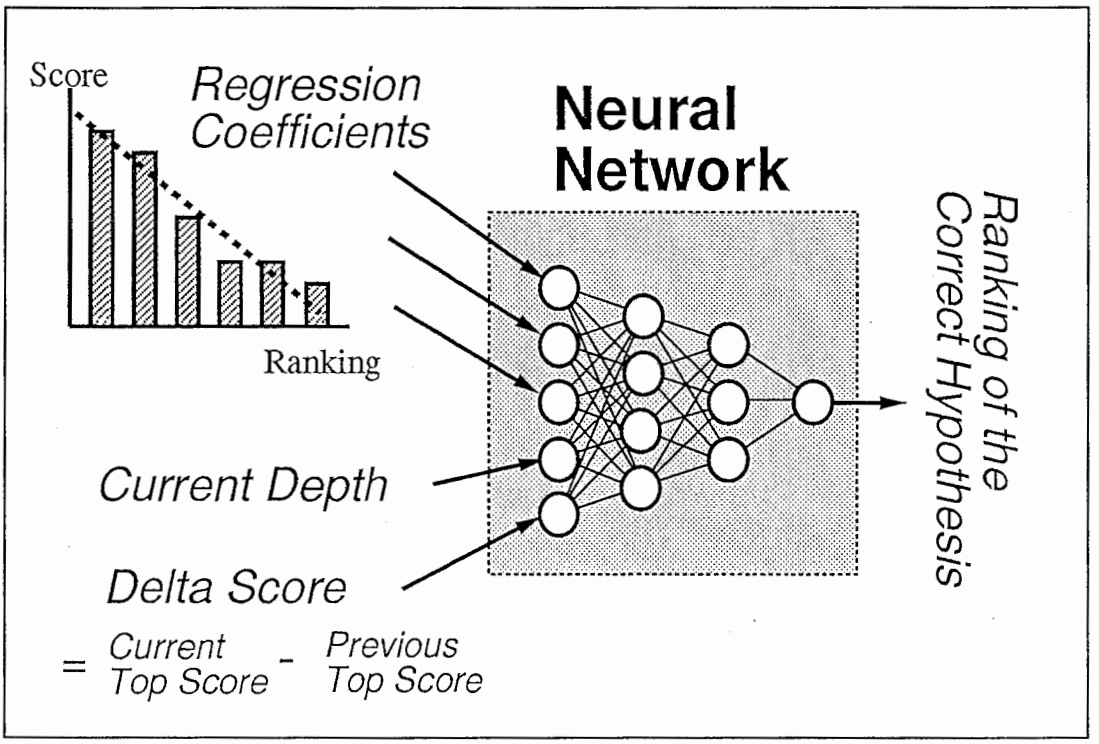

図 2.3: ニューラルネットワークの役割

```
$ cd CMHMM-LR/sampling
$ mksample.csh


::::::::::::::::::::::::
CMHMM-LR/sampling/mksample.csh
::::::::::::::::::::::::
# commands for learnibg by dcp2 (speaker dependent data)
# Usage: mksample.csh
set NAME = $1
set SPKRS = (FAF MAU FFS MHT FKS MMS FMS MMY FSU MNM)

touch temp.sample
foreach SPKR ($SPKRS)
 nawk -f mksample.awk \
   CMHMM-LR/Exe/features."$SPKR" >>  temp.sample
end

echo "SAMPLE FILE:"   >  temp.header
echo "6 inputs"       >> temp.header
echo "1 outputs"      >> temp.header
wc -l temp.sample > temp.wc
awk '{printf"%s patterns\n", $1}' temp.wc >> temp.header
cat temp.header temp.sample > samples.closed
rm -f temp.sample temp.header temp.wc

exit


::::::::::::::::::::::::
CMHMM-LR/sampling/mksample.awk
::::::::::::::::::::::::
# Count statistics for adaptive search (speaker dependent data)
# Usage: nawk -f mksample.awk <file name>
#
# $1: Correct or Error
# $2: depth
# $3: rank
# $4: phoneme string
# $5: prob(n,1) or prob(n,1)-prob(n,correct)
# $6: prob(n,1)-prob(n-1,1)
# $7: slope(n,1)
# $8: slope(n,1)-slope(n-1,1)
# $9: duration/mora
# $10: prob(n,1)-prob(n,4)
# $11: prob(n,1)-prob(n,8)
#
```

```
$1=="C" && $2!=1 && $3<11 && MABI==5 \
  {printf"1.0 %f %f %f %f %f %f\n", \
  $2/30,$6,$7,$10,$11,8*$3/10000+0.1; MABI=0} \
$1=="C" && $2!=1 && $3<11 && MABI<5 {MABI++}
$1=="C" && $2!=1 && 10<$3 && $3<51 \
  {for( i=0; i<2; i++) \
  printf"1.0 %f %f %f %f %f %f\n", \
    $2/30,$6,$7,$10,$11,8*$3/10000+0.1} \
$1=="C" && $2!=1 && 50<$3 && $3<101 \
  {for( i=0; i<10; i++) \
  printf"1.0 %f %f %f %f %f %f\n", \
    $2/30,$6,$7,$10,$11,8*$3/10000+0.1; \
  for( i=0; i<2; i++) \
  printf("1.0 %f %f %f %f %f %f\n", \
    ($2+rand()-0.5)/30, $6+((rand()-0.5)/1000), $7+((rand()-0.5)/1000), \
    $10+((rand()-0.5)/1000), $11+((rand()-0.5)/1000), \
    8*($3+($3-30)*rand()/9)/10000+0.1 )} \
$1=="C" && $2!=1 && 100<$3 && $3<151 \
  {for( i=0; i<20; i++) \
  printf"1.0 %f %f %f %f %f %f\n", \
    $2/30,$6,$7,$10,$11,8*$3/10000+0.1; \
  for( i=0; i<20; i++) \
  printf("1.0 %f %f %f %f %f %f\n", \
    ($2+rand()-0.5)/30, $6+((rand()-0.5)/1000), $7+((rand()-0.5)/1000), \
    $10+((rand()-0.5)/1000), $11+((rand()-0.5)/1000), \
    8*($3+($3-30)*rand()/9)/10000+0.1 )} \
$1=="C" && $2!=1 && 150<$3 && $3<1001 \
  {for( i=0; i<80; i++) \
  printf"1.0 %f %f %f %f %f %f\n", \
    $2/30,$6,$7,$10,$11,8*$3/10000+0.1; \
  for( i=0; i<80; i++) \
  printf("1.0 %f %f %f %f %f %f\n", \
    ($2+rand()-0.5)/30, $6+((rand()-0.5)/1000), $7+((rand()-0.5)/1000), \
    $10+((rand()-0.5)/1000), $11+((rand()-0.5)/1000), \
    8*($3+($3-30)*rand()/9)/10000+0.1 )} \
```

この例では先に示した9つの観測可能な特徴量の内、以下の5つを採用している。

- 回帰係数: a_1, b_2, b_1
- LR解析木の現在の深さ: n
- 現在の探索時点(深さn)の第1位仮説のスコアと直前の探索時点(深さ$n-1$)の第1位仮説のスコアとの差: $\Delta score$

サンプルはニューラルネットワークの入力・出力の範囲を考慮し、それぞれの特徴量毎に$[0,1]$に線形にスケーリングしている。さらに正解順位の低いサンプルの不足を補うために、特徴量毎

に微小のランダムノイズを加えることで見かけ上、それらのサンプルを人工的に増やしている。逆に、正解順位の高いサンプルは間引きしている。

【学習用サンプルファイルの例】

```
::::::::::::::::::::::::
samples.closed
::::::::::::::::::::::::
SAMPLE FILE:
6 inputs
1 outputs
1791 patterns
1.0 0.066667 -2.023000 -2.394000 -0.783000 3.908000 0.100800
1.0 0.100000 -0.559000 -1.363000 -0.382000 1.770000 0.100800
1.0 0.133333 1.114000 -0.783000 -0.360000 2.023000 0.103200
1.0 0.166667 0.003000 -1.039000 -0.241000 0.987000 0.100800
1.0 0.200000 -0.208000 -0.751000 -0.240000 1.185000 0.100800
1.0 0.233333 -0.506000 -0.377000 -0.200000 1.165000 0.100800
1.0 0.266667 0.507000 -0.708000 -0.324000 1.817000 0.100800
1.0 0.066667 -0.396000 -3.106000 -0.181000 -0.930000 0.100800
1.0 0.100000 -1.315000 -1.019000 -0.077000 -0.176000 0.126400
1.0 0.133333 -0.335000 -2.000000 -0.261000 0.438000 0.140000
1.0 0.166667 -1.016000 -0.652000 -0.247000 1.310000 0.198400
1.0 0.200000 -1.295000 -0.633000 -0.145000 0.584000 0.102400
1.0 0.066667 -0.007000 -3.800000 0.194000 -4.125000 0.100800
1.0 0.100000 0.242000 -2.556000 0.136000 -2.815000 0.100800
1.0 0.133333 -0.189000 -0.334000 -0.086000 0.375000 0.100800
1.0 0.166667 -0.345000 -0.483000 -0.150000 0.731000 0.100800
1.0 0.066667 -0.439000 -4.015000 -0.783000 2.739000 0.100800
1.0 0.100000 -0.322000 -1.451000 -0.621000 3.419000 0.103200
1.0 0.133333 -1.269000 -0.162000 -0.083000 0.483000 0.110400
1.0 0.166667 1.077000 -0.574000 -0.273000 1.548000 0.100800
1.0 0.200000 0.425000 -0.645000 -0.233000 1.212000 0.100800
1.0 0.233333 1.222000 -0.975000 -0.455000 2.568000 0.100800
1.0 0.066667 -0.364000 -2.750000 -0.987000 5.114000 0.102400
1.0 0.100000 -0.050000 -1.008000 -0.253000 1.097000 0.100800
1.0 0.133333 1.114000 -1.546000 -0.495000 2.450000 0.100800
1.0 0.166667 -0.448000 -0.901000 -0.193000 0.739000 0.100800

  …… (中略) ……

1.0 0.133333 -0.113000 -0.688000 -0.290000 1.589000 0.108800
1.0 0.166667 -0.201000 -0.525000 0.048000 -0.726000 0.100800
1.0 0.200000 1.078000 -0.670000 -0.313000 1.769000 0.104000
1.0 0.233333 -0.344000 -0.563000 -0.209000 1.098000 0.105600
1.0 0.266667 0.220000 -1.270000 -0.371000 1.756000 0.100800
1.0 0.066667 2.622000 -2.154000 -0.430000 1.541000 0.100800
1.0 0.100000 3.353000 -1.640000 -0.564000 2.872000 0.100800
1.0 0.133333 1.727000 -1.102000 -0.400000 2.084000 0.100800
1.0 0.166667 0.376000 -2.093000 -0.640000 3.096000 0.102400
1.0 0.200000 1.803000 -1.052000 -0.402000 2.136000 0.100800
1.0 0.066667 0.951000 -1.548000 -0.244000 0.640000 0.101600
1.0 0.100000 -0.518000 -0.870000 -0.156000 0.494000 0.101600
1.0 0.133333 -0.001000 -0.535000 -0.223000 1.216000 0.101600
1.0 0.066667 -1.042000 -0.980000 -0.305000 1.489000 0.100800
```

```
1.0 0.100000 1.026000 -1.054000 -0.550000 3.195000 0.100800
1.0 0.133333 0.294000 -0.928000 -0.262000 1.213000 0.100800
1.0 0.166667 -0.982000 -0.443000 -0.169000 0.896000 0.100800
1.0 0.200000 0.085000 -0.706000 -0.117000 0.336000 0.100800
1.0 0.233333 -0.072000 -0.480000 -0.201000 1.102000 0.100800
```

2.4.2 ネットワークの構築

構造を決めるにはニューラルネット・ワークベンチシステム “nnwb” を用いる。このシステムの取り扱い説明については、文献 [12] を参照してください。

ここでは 4 層のパーセプトロン型のニューラルネットワークである。入力ユニットは 5 個、第 1 中間層は 4 ユニット、第 2 中間層は 3 ユニットからなる。出力ユニットは 1 個で、正解仮説の順位を出力するように学習する。

以下の 3 つのファイルを作成する。

```
CMHMM-LR/NN/weight
CMHMM-LR/NN/para
CMHMM-LR/NN/net
```

2.4.3 学習

ニューラルネットワーク学習プログラム “DCP2” を用いる。このシステムの取り扱い説明については、文献 [11] を参照してください。ディレクトリ CMHMM-LR/regress で学習用シェルスクリプト train.csh を用いる。

【使用例】

```
$ train.csh
```

2.4.4 単体の評価

学習と同様、ニューラルネットワーク学習プログラム “DCP2” を用いる。よって、文献 [11] を参照してください。ディレクトリ CMHMM-LR/regress で評価用シェルスクリプト eval.csh を用いる。結果は xgraph2 等で見ることができる。

【使用例】

```
$ eval.csh
$ xgraph2 -P -nl in+out.xg


::::::::::::::::::::::::
CMHMM-LR/regress/eval.csh
::::::::::::::::::::::::
# commands for evaluation by dcp2 on HP9000 (speaker dependent data)
# Usage: eval.csh

cp -f samples.open sample
rm -f out.tmp out in in+out in+out.disp
cd ..
dcp2_hp -rc 1 NN > NN/out.tmp
```

```
cd NN
awk 'NF==4 && $3==0 {print $4}' out.tmp > out
awk 'NF>6 {print $NF}' sample > in
wc -l in out
paste in out > in+out

awk '{print 10000*($1-0.1)/8,10000*($2-0.1)/8}' in+out > in+out.disp
cat header.xg in+out.disp > in+out.xg
rm -f in+out.disp

exit
```

2.4.5 文節音声認識での評価

HMM-LR を用いて文節音声認識実験を行なう。HMM-LR 連続音声認識システムの技術的内容については、文献 [2] を参照してください。プログラムの取り扱い説明については文献 [4, 10] と同じ仕様に基づいている。ただし、-B オプションは最大ビーム幅の拡大率／縮小率 (%) を表している。100% が 720 に相当する。-a オプションは式 (2.3) における $margin$ を表している。$\varphi_N(O_1(n), O_2(n), \cdots, O_m(n)) + margin$ の値は抽出結果の例中、HMM の尤度計算時間の表示の下に印字されている。ニューラルネットワークのパラメータファイルは CMHMM-LR/NN に格納されているものとする。

【使用例】

```
$ cd CMHMM-LR/Exe
$ ../Src/adbeam/main_NN \
  ../DATA/MSH_List_SB3 \
  -g ../GRAMMAR/sp2 \
  -m ../HMM/MSH/model_list \
  -D ../DATA/para_list.MSH \
  -P 5 -M 1 -B 60 -b 64 -c 3000 -a 30 \
  -s 2 -S 3 -W 0 -A 7 -v 1.2 -f 0 -p 0 \
  | tee result.MSH
```

【抽出結果の例】

```
::::::::::::::::::::::::
CMHMM-LR/Exe/result.MSH
::::::::::::::::::::::::
(001)  147120 182   |daiikkai|
Making likelihood map: CPU-time = 18451 msec, Elapsed-time = 19 sec.
 1 : 519 2 : 151 3 : 189 4 : 206 5 : 215 6 : 245 7 : 265 8 : 218 9 : 334
10 : 305 11 : 327 12 : 321 13 : 347 14 : 360 15 : 356 16 : 354 17 : 360 18
: 384 19 : 230
************************************************************
Parsing time: CPU-time = 18479 msec, Elapsed-time = 20 sec.
Total-verify = 5028  Depth = 20
  1: daiiQ-kai                (prob = -46.32508)
  2: daiiQ-kai-ni             (prob = -45.93980)
  3: daiiQ-kai-e              (prob = -45.87721)
```

```
  4: daizyuQ-kai                (prob = -45.80021)
  5: daitai                     (prob = -45.76401)
**************************************************************

(002)  147382 170   |tsuuyaku|
Making likelihood map: CPU-time = 17061 msec, Elapsed-time = 17 sec.
 1 : 519 2 : 280 3 : 138 4 : 139 5 : 220 6 : 242 7 : 256 8 : 250 9 : 258
10 : 280 11 : 291 12 : 333 13 : 360 14 : 345 15 : 314 16 : 375 17 : 308 18
: 369
**************************************************************
Parsing time: CPU-time = 16877 msec, Elapsed-time = 19 sec.
Total-verify = 4951  Depth = 19
  1: tsuuyaku                   (prob = -45.65362)
  2: tsuuyaku-e                 (prob = -45.02155)
  3: tsuuyaku-o                 (prob = -44.97915)
  4: tsuuyaku-ka                (prob = -44.71666)
  5: tsuuyaku-ga                (prob = -44.68537)
**************************************************************

(003)  147674 123   |deNwa|
Making likelihood map: CPU-time = 12331 msec, Elapsed-time = 12 sec.
 1 : 519 2 : 243 3 : 233 4 : 281 5 : 269 6 : 258 7 : 303 8 : 294 9 : 310
10 : 307 11 : 304 12 : 323 13 : 335
**************************************************************
Parsing time: CPU-time = 11663 msec, Elapsed-time = 13 sec.
Total-verify = 4665  Depth = 14
  1: deNwa                      (prob = -47.90800)
  2: zeN-wa                     (prob = -47.62052)
  3: neN-wa                     (prob = -47.54892)
  4: deNwa-wa                   (prob = -47.54102)
  5: deNwa-o                    (prob = -47.26934)
**************************************************************

(004)  148002 160   |kokusai|
Making likelihood map: CPU-time = 16069 msec, Elapsed-time = 16 sec.
 1 : 519 2 : 69 3 : 131 4 : 214 5 : 294 6 : 204 7 : 115 8 : 319 9 : 302 10
: 221 11 : 303 12 : 331 13 : 299 14 : 333 15 : 312 16 : 352 17 : 358
**************************************************************
Parsing time: CPU-time = 15459 msec, Elapsed-time = 17 sec.
Total-verify = 5051  Depth = 18
  1: kokusai                    (prob = -44.51477)
  2: kokusai-ni                 (prob = -44.06600)
  3: kokusai-e                  (prob = -43.96870)
  4: kokusai-o                  (prob = -43.81251)
  5: kokusai-ne                 (prob = -43.57242)
**************************************************************

(005)  148302 152   |kaigini|
Making likelihood map: CPU-time = 15291 msec, Elapsed-time = 16 sec.
 1 : 520 2 : 127 3 : 209 4 : 206 5 : 220 6 : 297 7 : 222 8 : 287 9 : 280
10 : 290 11 : 303 12 : 276 13 : 335 14 : 327 15 : 306 16 : 315
**************************************************************
Parsing time: CPU-time = 15104 msec, Elapsed-time = 16 sec.
Total-verify = 4976  Depth = 17
```

```
  1: kaigi-ni                    (prob = -43.00225)
  2: kaigizyou-ni                (prob = -42.85176)
  3: kaigi-yori                  (prob = -42.65457)
  4: kaizyou-ni                  (prob = -42.56547)
  5: kai-ni                      (prob = -42.42308)
*************************************************************
```

……（以下省略）……

2.5 重回帰分析

重回帰分析の構築・学習・評価はディレクトリ CMHMM-LR/regress で行なう。本ツールで用いるプログラムはソフトウェアパッケージ “Numerical Recipes in C” を DECstation 用に修正したものである。

2.5.1 観測量のフォーマット化

抽出結果ファイルを重回帰分析で学習するために、フォーマット化された学習用サンプルファイルに変換する。作成にあたっては、ディレクトリ CMHMM-LR/regress の下で以下のツールを使う。このフォーマットはニューラルネットワーク学習プログラム “DCP2”[11] に準じている。

mksample.awk 学習サンプル抽出 AWK プログラム
mksample.csh 学習サンプル抽出シェルスクリプト

【使用例】

```
$ cd CMHMM-LR/regress
$ mksample.csh


::::::::::::::::::::::::
CMHMM-LR/regress/mksample.csh
::::::::::::::::::::::::
# commands for learnibg by regression analysis
# Usage: mksample.csh
set SPKRS = (FAF MAU FFS MHT FKS MMS FMS MMY FSU MNM)

touch samples.closed
foreach SPKR ($SPKRS)
  nawk -f ../sampling/mksample.awk \
    CMHMM-LR/Exe/features."$SPKR"  >> samples.closed
end
```

この例でもニューラルネットワークと同様、疑似増加および間引きされた 5 つの観測可能な特徴量を採用している。ただし、スケーリングはしていない。

【学習用サンプルファイルの例】

```
::::::::::::::::::::::::
samples.closed
::::::::::::::::::::::::
```

```
C  7   1 iQ-ka  0.000 -0.450 -1.016  0.403 26.80 -0.125  0.208 45.460 1329
C 3 40 tsuuy  2.232 -1.317 -1.022  1.391 23.00  0.078 -1.637 41.854 1158
C 3 39 tsuuy  2.232 -1.319 -1.022  1.395 23.00  0.080 -1.640 41.854 1154
C 2 39 tsuuy  2.232 -1.319 -1.023  1.396 23.00  0.081 -1.636 41.857 1160
C 4 12 suuya  1.557 -0.518 -1.407 -0.385 25.33 -0.426  2.597 41.019 1718
C 4 11 suuya  1.557 -0.517 -1.409 -0.387 25.33 -0.426  2.596 41.020 1720
C 4 12 suuya  1.557 -0.514 -1.405 -0.382 25.33 -0.427  2.596 41.016 1721
C  2   1 de     0.000 -2.329 -3.310  3.063 26.00 -0.560  2.083 49.555 183
C 4 12 deNw  1.933 -0.826 -1.785 -0.015 33.50 -0.407  2.078 47.497 1884
C 3 11 deNw  1.933 -0.821 -1.789 -0.018 33.50 -0.405  2.077 47.500 1886
C 4 11 deNw  1.933 -0.828 -1.782 -0.018 33.50 -0.412  2.083 47.495 1879
C 5 14 deNwa  0.912 -1.689 -0.820  0.965 23.33 -0.207  1.134 45.873 1817
C 4 14 deNwa  0.912 -1.688 -0.824  0.966 23.33 -0.207  1.131 45.873 1816
C 5 13 deNwa  0.912 -1.690 -0.821  0.960 23.33 -0.207  1.136 45.869 1816
C  5   1 kokus  0.000  0.347 -0.784  1.043 25.00 -0.290  1.927 41.855 1878
C  4   6 kai-g  0.026 -1.149 -0.378  0.435 17.50  0.079 -1.060 41.247 1988
C  3   6 saN    0.942 -1.859 -1.585  2.901 24.50 -0.112 -0.425 45.018 1158
C 4 25 saN-k  1.708 -0.335 -0.750  0.835 27.50 -0.320  2.228 44.179 1863
C 4 24 saN-k  1.708 -0.337 -0.753  0.833 27.50 -0.316  2.223 44.181 1861
C 4 25 saN-k  1.708 -0.331 -0.753  0.838 27.50 -0.317  2.227 44.181 1866
C 5 20 aN-ka  0.653 -1.536 -0.464  0.286 25.00 -0.084  0.342 43.196 1665
C 4 19 aN-ka  0.653 -1.540 -0.460  0.289 25.00 -0.088  0.340 43.199 1662
C 5 20 aN-ka  0.653 -1.539 -0.459  0.284 25.00 -0.081  0.346 43.200 1669
C 6 14 -ka-n  0.562 -0.629 -0.463  0.001 30.67 -0.072  0.230 42.641 1427
C 5 13 -ka-n  0.562 -0.633 -0.463 -0.002 30.67 -0.070  0.231 42.638 1426
C 6 13 -ka-n  0.562 -0.629 -0.466 -0.002 30.67 -0.071  0.235 42.641 1429
C  5   1 ourok  0.000 -0.509 -1.021 -0.053 24.67 -0.327  2.047 42.624 1980
C 3 91 go-k  3.884 -1.299 -2.003  0.400 11.00 -0.386  1.692 46.253 1158
C 3 90 go-k  3.884 -1.299 -2.003  0.396 11.00 -0.388  1.688 46.256 1160
C 2 91 go-k  3.884 -1.303 -1.998  0.401 11.00 -0.385  1.693 46.256 1154
C 2 90 go-k  3.884 -1.297 -2.005  0.401 11.00 -0.388  1.691 46.254 1157
C 3 90 go-k  3.884 -1.303 -2.005  0.395 11.00 -0.390  1.691 46.254 1156
C 3 90 go-k  3.884 -1.298 -2.003  0.397 11.00 -0.389  1.693 46.251 1161
C 3 91 go-k  3.884 -1.295 -2.002  0.400 11.00 -0.384  1.694 46.251 1154
```

…… (中略) ……

```
C 7 138 iko-m  1.603 -0.969 -0.826  0.247 26.00 -0.258  1.605 44.051 1374
C 6 137 iko-m  1.603 -0.971 -0.820  0.248 26.00 -0.263  1.607 44.058 1377
C 7 138 iko-m  1.603 -0.978 -0.822  0.243 26.00 -0.256  1.608 44.049 1378
C 6 137 iko-m  1.603 -0.977 -0.824  0.243 26.00 -0.254  1.609 44.056 1372
C 6 138 iko-m  1.603 -0.972 -0.820  0.243 26.00 -0.260  1.612 44.057 1379
C 6 138 iko-m  1.603 -0.978 -0.820  0.248 26.00 -0.259  1.611 44.050 1378
C 6 138 iko-m  1.603 -0.977 -0.826  0.246 26.00 -0.259  1.610 44.056 1377
C 6 138 iko-m  1.603 -0.978 -0.821  0.242 26.00 -0.256  1.610 44.055 1380
C 6 138 iko-m  1.603 -0.976 -0.824  0.249 26.00 -0.259  1.609 44.055 1380
C 6 138 iko-m  1.603 -0.976 -0.824  0.243 26.00 -0.264  1.607 44.049 1381
C 7 138 iko-m  1.603 -0.976 -0.822  0.247 26.00 -0.261  1.609 44.055 1377
C  8   1 ko-mi  0.000 -0.645 -0.604  0.220 32.20 -0.139  0.710 43.605 1432
C 6 24 hi-no  0.995 -0.053 -0.540 -0.124 24.75 -0.178  1.123 46.571 1427
C 6 23 hi-no  0.995 -0.049 -0.536 -0.125 24.75 -0.177  1.122 46.567 1429
C 6 24 hi-no  0.995 -0.055 -0.545 -0.121 24.75 -0.183  1.120 46.567 1422
C  2   1 ke     0.000 -0.439 -2.753  0.305 28.00 -0.049 -2.059 48.916 183
C 3 11 yo-r  0.707 -1.090 -0.697  4.084 21.00 -0.157  0.789 45.182 1158
```

```
C 3 11 yo-r  0.707 -1.087 -0.696  4.087 21.00 -0.157  0.788 45.184 1161
C 2 10 yo-r  0.707 -1.088 -0.699  4.088 21.00 -0.152  0.793 45.186 1159
C 4 15 yoro  1.001 -0.929 -0.570  0.127 10.50 -0.136  0.720 44.121 1723
C 4 15 yoro  1.001 -0.926 -0.568  0.127 10.50 -0.140  0.717 44.125 1727
C 4 15 yoro  1.001 -0.932 -0.569  0.130 10.50 -0.133  0.724 44.124 1724
C  7   2 shi-k  0.128  0.057 -1.004 -0.022 34.00 -0.187  0.789 44.434 1355
C  5   1 onega  0.000  0.889 -1.352 -0.322 34.00 -0.486  3.185 44.401 1741
C  2   1 ag     0.000  1.514 -3.127  1.621 26.00 -0.688  3.410 40.635 183
C  7   1 ma-su  0.000 -0.060 -1.275  0.100 33.00 -0.331  1.854 42.901 1483
```

2.5.2 学習

ディレクトリ CMHMM-LR/regress で学習プログラム main を用いる。学習の結果得られた重回帰係数はファイル coef.data に格納しておく。

【使用例】

```
$ main samples.closed > coef.data
```

2.5.3 単体の評価

ディレクトリ CMHMM-LR/regress で学習プログラム test、評価用シェルスクリプト eval.csh を用いる。結果は xgraph2 等で見ることができる。

【使用例】

```
$ eval.csh
$ xgraph2 -P -nl result.open
```

```
::::::::::::::::::::::::
CMHMM-LR/NN/eval.csh
::::::::::::::::::::::::
set SPKRS = (MSH FKM MTK FYM)
touch result.open
foreach SPKR ($SPKRS)
  test CMHMM-LR/Exe/features."$SPKR" >> result.open
end

exit
```

2.5.4 文節音声認識での評価

HMM-LR を用いて文節音声認識実験を行なう。仕様はニューラルネットワークの場合と同じなので説明は省略する。重回帰係数は CMHMM-LR/regress/coef.data に格納されているものとする。

【使用例】

```
$ cd CMHMM-LR/Exe
$ ../Src/adbeam/main_reg \
  ../DATA/MSH_List_SB3 \
```

```
-g ../GRAMMAR/sp2 \
-m ../HMM/MSH/model_list \
-D ../DATA/para_list.MSH \
-P 5 -M 1 -B 60 -b 64 -c 3000 -a 30 \
-s 2 -S 3 -W 0 -A 7 -v 1.2 -f 0 -p 0 \
| tee result.MSH
```

第 3 章

A^* アルゴリズムに基づく探索

best-first 探索において部分仮説 n の評価値を次の式で置き換えたものである。

$$\hat{f}(n) = g(n) + \hat{h}(n) \tag{3.1}$$

ここで $g(n)$ は部分仮説 n の HMM と入力音声の対数尤度であり、$\hat{h}(n)$ は未探索部分の入力音声の対数尤度の推定値である。したがって評価値も真の値 $f(n)$ に対する推定値となる。最適解が保証されるためには未探索部分の入力音声の真の対数尤度 $h(n)$ に対して認容可能性：$\hat{h}(n) \geq h(n)$ が成立しなければならない。また、$\hat{h}(n)$ の推定精度は探索効率に大きく影響する。$\hat{h}(n)$ はできる限り $h(n)$ に近い方が、展開するノードの数が少なくてすむ。

3.1 ヒューリスティック関数 $\hat{h}(n)$

HMM-LR において正確な $h(n)$ を求めるには、逆向きに LR パーザを動作させることにより算出することは可能であるが、仮説毎に別々にスタックを用意する必要があり、また、前向きに

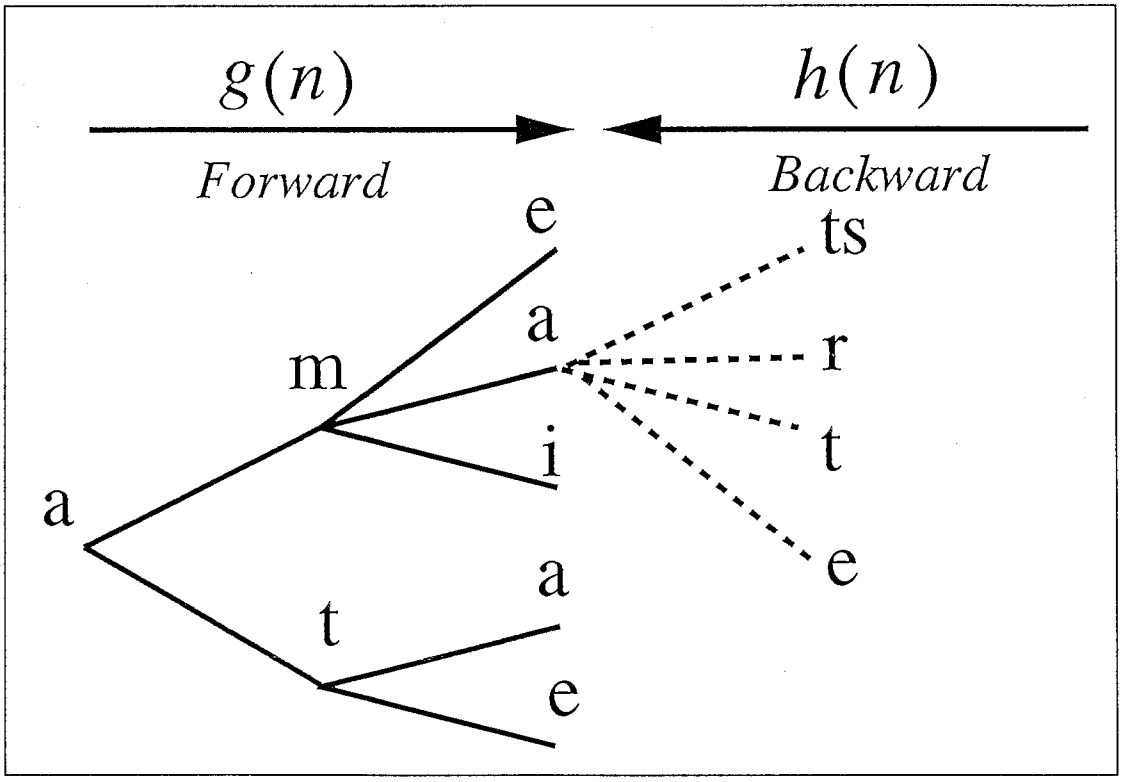

図 3.1: HMM-LR 探索木と A^* アルゴリズム

展開してきたノードと後向きに展開してきたノードとの接続可能性を判定しなければならず、実際には不可能である。

われわれは計算の効率を考慮し、すべての音素モデル同士が接続可能として、$\hat{h}(n)$ とした。$\hat{h}(n)$ は探索の前処理として、探索とは逆方向に、文の終端からし始端に向かって Viterbi Algorithm で求める。ただしこのままでは $h(n)$ に比べ $\hat{h}(n)$ は非常に大きな値となり、かつすべての部分仮説に対し、同じ時刻で $\hat{h}(n)$ は同じ値になってしまうため、認容可能性は満たしているもののヒューリスティック力が弱く盲目的探索に陥る危険がある。そこで次の2つの方式を導入した。

I. $\hat{h}(n)$ の精度向上 (→ 図3.1)

現時点のノード n から LR テーブルを1つ先読みして、そのノードに接続可能な音素候補の集合 P_n を求める。ノード n の時刻 t における $\hat{h}_n(t)$ は、P_n に含まれるすべての音素 p について $\hat{h}_n(p,t)$ の最大値とする。探索時に部分仮説 n の評価値 $\hat{f}(n)$ は、

$$\hat{f}(n) = \max_{t_1 \le t \le t_2} (g_n(t) + \max_{p \in P_n}(\hat{h}_n(p,t))). \tag{3.2}$$

II. 最適条件の緩和

$\hat{h}_n(t)$ に探索が深くなるにつれて寄与が小さくなるように動的重み付けをして最適条件を緩和した [13]。認容可能性が成立しないこともあり得るが、今回の実験では高速性を重視して採用した。

$$\hat{f}(n) = \max_{t_1 \le t \le t_2} (g_n(t) + w(t)\hat{h}_n(t)). \tag{3.3}$$

(ここで $w(t) = 0.8 - 0.1t/T$, T は入力音声長)

実験では音素照合回数の上限を 10,000 に設定し、それを越えたときは探索を打ち切っている。

本ソフトウェアは検討段階のものであるため、$\hat{h}(n)$ は文節音声認識に先立ち、予め算出してファイル h_hat_XXX.data に格納しておく形式になっている。ここで、"XXX" は話者名である。

【使用例】

```
$ cd CMHMM-LR/Exe
$ ../heuristic/h_func \
  ../DATA/MSH_List_SB3 \
  -g ../GRAMMAR/sp2 \
  -m ../HMM/MSH/model_list \
  -D ../DATA/para_list.MSH \
  -P 5 -M 1 -B 600 -b 64 -c 3000 \
  -s 2 -S 3 -W 0 -A 7 -v 1.2 -f 0 -p 0 \
  > h_hat_MSH.log

$ awk '$1=="@" {F=1;print $2,$3,$4} $1!="@" && F==1 {print $0}' \
        h_hat_MSH.log > h_hat_MSH.data
$ rm -f h_hat_MSH.log
```

【抽出結果の例】

```
::::::::::::::::::::::::
CMHMM-LR/Exe/h_hat_MSH.data
::::::::::::::::::::::::
# 1 daiikkai
t= 189
max= 0.000000
```

```
t= 188
max= 0.000000
t= 187
p= 22.036298
p= 24.494575
p= 77.402130
p= 55.816109
p= 66.745463
p= 22.036298
p= 61.451727
p= 101.223139
p= 58.854657
p= -9.328734
p= 15.963902
p= 16.694565
p= 31.355532
p= 132.368482
p= 61.233438
p= 30.522674
p= 131.280536
p= 61.209440
p= -42.890156
p= -25.255237
p= -2.210289
p= -50.831958
p= 13.987001
p= -3.832808
p= 51.064747
p= -18.450178
p= -11.703315
p= 54.179191
p= 13.987001
p= -59.831708
p= 17.938503
p= 130.028599
p= -82.770862
p= 100.472876
p= 130.028599
p= -60.941853
p= 58.546573
p= 59.167142
p= 12.157692
p= -59.915504
p= 38.569072
p= 4.718264
p= 41.480526
p= 7.625719
p= 52.837358
p= 37.362032
p= 22.036298
p= 24.494575
p= 22.036298
p= 61.451727
p= 60.235188
p= 58.854657
p= -22.057797
p= 16.694565
p= 61.233438
p= -32.882856
p= 61.209440
p= 13.987001
p= 13.987001
p= 17.938503
p= 8.352382
p= 130.028599
p= 59.167142
p= 4.718264
max= 132.368482
t= 186
p= 60.088296
p= 62.468677
p= 108.404480
p= 70.270187
p= 100.203490
p= 60.088296
p= 105.017349
p= 151.319334
```

```
p= 102.514374
p= -5.435628
p= 25.431528
p= 39.820309
p= 53.155542
p= 187.164942
p= 89.375331
p= 61.166742
p= 182.356682
p= 99.638518
p= -32.859157
p= 2.974551
p= 8.850257
p= -53.597420
p= 37.196440
p= 9.182641
p= 79.944003
p= 0.755062
p= 1.702415
p= 79.809710
p= 37.196440
p= -57.667517
p= 56.732963
p= 183.727514
p= -70.718264
p= 138.709965
p= 183.727514
p= -53.097545
p= 83.627519
p= 97.657517
p= 36.940953
p= -52.327984
p= 56.055497
p= 26.272886
p= 57.225939
p= 29.075046
p= 77.197340
p= 70.347483
p= 60.088296
p= 62.468677
p= 60.088296
p= 105.017349
p= 95.426529
p= 102.514374
p= -12.721164
p= 39.820309
p= 89.375331
p= -28.242888
p= 99.638518
p= 37.196440
p= 37.196440
p= 56.732963
p= 46.665667
p= 183.727514
p= 97.657517
p= 26.272886
max= 187.164942
t= 185
p= 95.700715
p= 98.003200
p= 145.233138
p= 144.515574
p= 159.839208
p= 123.463627
p= 161.447028
p= 202.494475
p= 182.670967
p= 98.214389
p= 132.116494
p= 125.275036
p= 145.824209
p= 244.982951
p= 211.133543
p= 171.564122
p= 273.680916
p= 210.724864
p= 100.703065
```

```
p= 131.605820
p= 155.519824
p= 89.438628
p= 153.712814
p= 151.954602
p= 200.462677
p= 164.515574
p= 150.391581
p= 200.546773
p= 153.712814
p= 73.968102
p= 176.814159
p= 278.559072
p= 80.907855
p= 243.978001
p= 278.559072
p= 102.199290
p= 203.869207
p= 208.682566
p= 162.627169
p= 93.735513
p= 191.714914
p= 146.922554
p= 174.558872
p= 149.830009
p= 188.302385
p= 179.275136
p= 178.429979
p= 180.888256
p= 178.429979
```

…… (以下省略) ……

3.2 文節音声認識での評価

HMM-LR を用いて文節音声認識実験を行なう。通常のビームサーチの場合と異なるものについてのみ仕様を説明する。-B オプションはスタックサイズを表している。-H オプションはヒューリスティック関数 $\hat{h}(n)$ の格納ファイルを指定する。-h オプションは式 (3.3) における重み係数 $w(t)$ の初期値 (標準的には 0.8 に設定) を表している。-n オプションは N-best であって、最終的に第何位まで算出するかを表している。最適条件を緩和しているため正確な N-best にはなっていなく、第 1 位から順に求まるとは限らないことに注意。

【使用例】

```
$ cd CMHMM-LR/Exe
$ ../Src/A-star/main \
  ../DATA/MSH_List_SB3 \
  -g ../GRAMMAR/sp2 \
  -m ../HMM/MSH/model_list \
  -D ../DATA/para_list.MSH \
  -P 5 -M 1 -B 900 -b 64 -c 3000 \
  -s 2 -S 3 -W 0 -A 7 -v 1.2 -f 0 -p 0 -h 0.8 -n 300 \
  -H h_hat_MSH.data \
  | tee result.MSH
```

参考文献

[1] 花沢利行、川端豪、鹿野清宏：“HMM による音韻認識実験の結果,” ATR Technical Report, TR-I-0147 (1990.2).

[2] 北研二、川端豪、斉藤博昭: “HMM 音韻認識と拡張 LR 構文解析法を用いた連続音声認識,” 情報処理学会論文誌, Vol.31, No.3, pp.472–479 (1990.3).

[3] 北研二、川端豪、森元逞: “HMM-LR 連続音声認識システムにおける計算量削減のための一検討,” 音講論集, 3-6-4, pp.77–78 (1989.3).

[4] 北研二: “HMM-LR ユーザーズ・マニュアル,” ATR Technical Report, TR-I-0246 (1992.3).

[5] 山口耕市、嵯峨山茂樹: “混合連続分布型 HMM を用いた HMM-LR 連続音声認識,” 音講論集, 1-P-5, pp.113–114 (1992.3).

[6] 山口耕市, 嵯峨山茂樹, 北研二, Frank K. Soong: “HMM-LR 連続音声認識における A^* アルゴリズムを用いた探索手法の検討,”

日本音響学会平成 4 年度春季研究発表会講演論文集, 3-1-9, pp.87–88 (1992.3).

[7] K. Yamaguchi, K. Kita, S. Sagayama and F. K. Soong: “Continuous Mixture HMM-LR Using the A^* Algorithm for Continuous Speech Recognition,” *Proc. ICSLP-92*, We.sAM.1.2, pp.301–304 (1992.10).

[8] 山口耕市, 嵯峨山茂樹: “ニューラルネットワークを用いた適応的ビームサーチによる HMM-LR 連続音声認識,” 音学講論, 3-7-2, pp.165–166 (1992.10).

[9] 山口耕市, 嵯峨山茂樹: “ニューラルネットワークによる学習可能な探索手法を用いた連続音声認識,” 信学技報, SP92-123, pp. 1–7 (1993.1).

[10] 山口耕市: “混合連続分布型 HMM-LR ユーザーズマニュアル,” ATR Technical Report, TR-I-0364 (1993.3).

[11] 中村雅己, 鹿野清宏: “ニューラルネットにおけるバックプロパゲーション学習の効率化方法,” ATR Technical Report, TR-I-0119 (1989.10).

[12] 福沢圭二, 中村雅己: “ニューラルネット・ワークベンチシステム – 操作マニュアル –,” ATR Technical Report, TR-I-0192 (1991.2).

[13] A. Barr, E. A. Feigenbaum 著 (田中、淵監訳)：“人工知能ハンドブック,” 第 II 章, 共立出版 (1983).