東京大学 工学部 計数工学科/物理工学科

応用音響学：音声認識 (6) 言語モデルと探索

嵯峨山 茂樹 ＜sagayama@hil.t.u-tokyo.ac.jp＞
東京大学 工学部 計数工学科 http://hil.t.u-tokyo.ac.jp/

連続音声認識の構成/原理

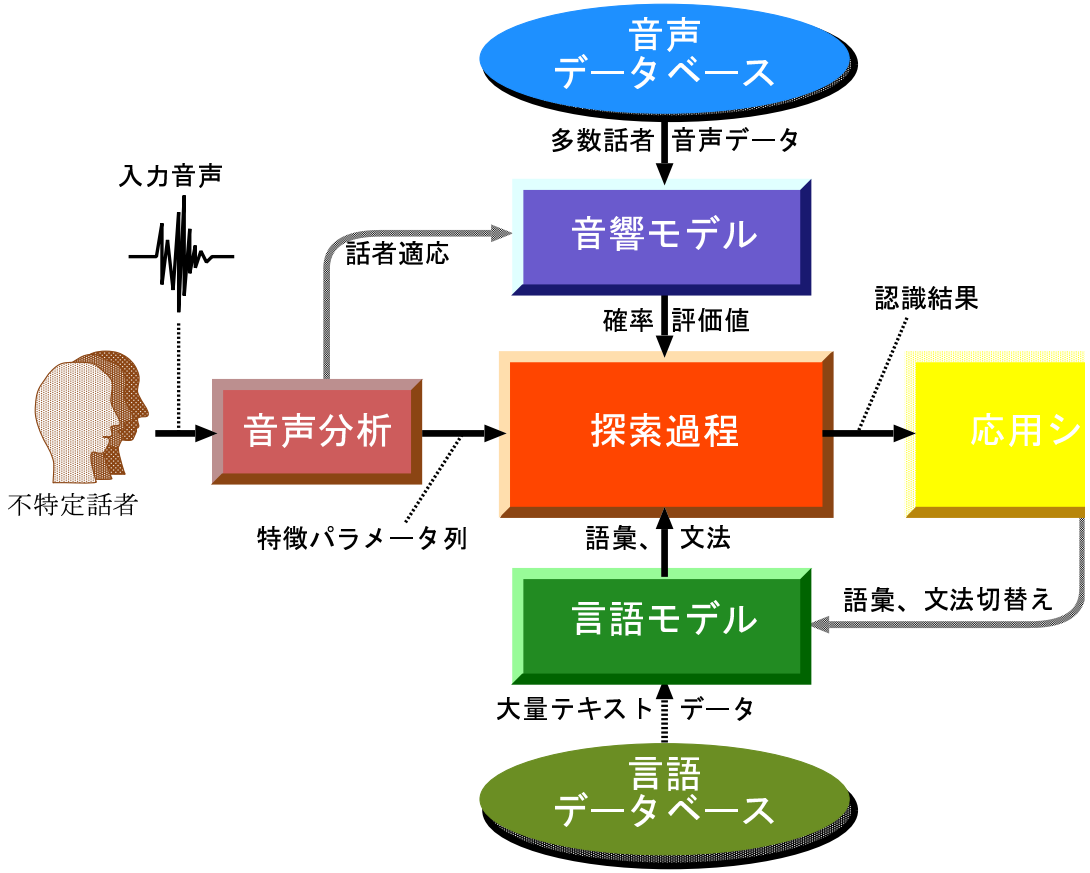


図1. 連続音声認識の構成要素。音声分析により音声特徴を抽出し、語彙や文法によって音素の並びを規定する言語モデルのもとで、音素を確率モデルにより表現する音響モデルによって入力の確率評価をして、事後確率最大の経路を効率良く探索する。

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

言語モデルと探索手法

- いままでは、音声信号波形から一度、母音や子音などの音素系列に変換し、その誤りを文法を適用して修正するボトムアップタイプが主流だったが、この方法では音素認識と文法処理の間の情報の欠落が大きく、性能に限界があった。近年は、音素モデルと言語モデルを緊密に連係した方法が主流となりつつある。
- 簡単な適用対象の場合、文法を有限状態オートマトン (FSA) で記述することが多い。状態間遷移に単語を対応させ、状態遷移ネットワーク全体で文法を記述するものである。ネットワーク中の一つの経路は、一つの文に対応する。HMM サブワードモデルはこの中に埋め込まれ、音声認識は、入力音声を生成する最もよいモデルになっている経路を探索する問題になる。これは、基本的には全探索によらねば解けない問題である。それを効率良く実行するアルゴリズムとして one-pass アルゴリズムあるいは level-building アルゴリズムがある。さらに、あまり有望でない経路は途中で計算を打ち切るという、「枝刈り」の手法が併用される。最もよい経路のみでなく、上位複数個の経路も捜したいことが多いので、N-best アルゴリズムと呼ばれるアルゴリズムも用いられる。
- 少し複雑な対象には、より記述能力の高い「文脈自由文法」を構文規則記述に用いる。その場合の解探索は簡単な問題ではない。各種のパーザ (構文解析) が開発されているが、その中でも一般化 LR 構文解析法を用いると効率が高い。連続音声の認識では、連続音声は音素が時間的に連結されたものなので、文法で許される範囲でいろいろな音素モデルを接続して、入力された音声と照合する。「LR 構文解析法」は文法により予測される音素を連結して照合するので無駄な接続を行なわない効率の良い音声認識を行なう方式である。

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

確率論から捉えた連続音声認識

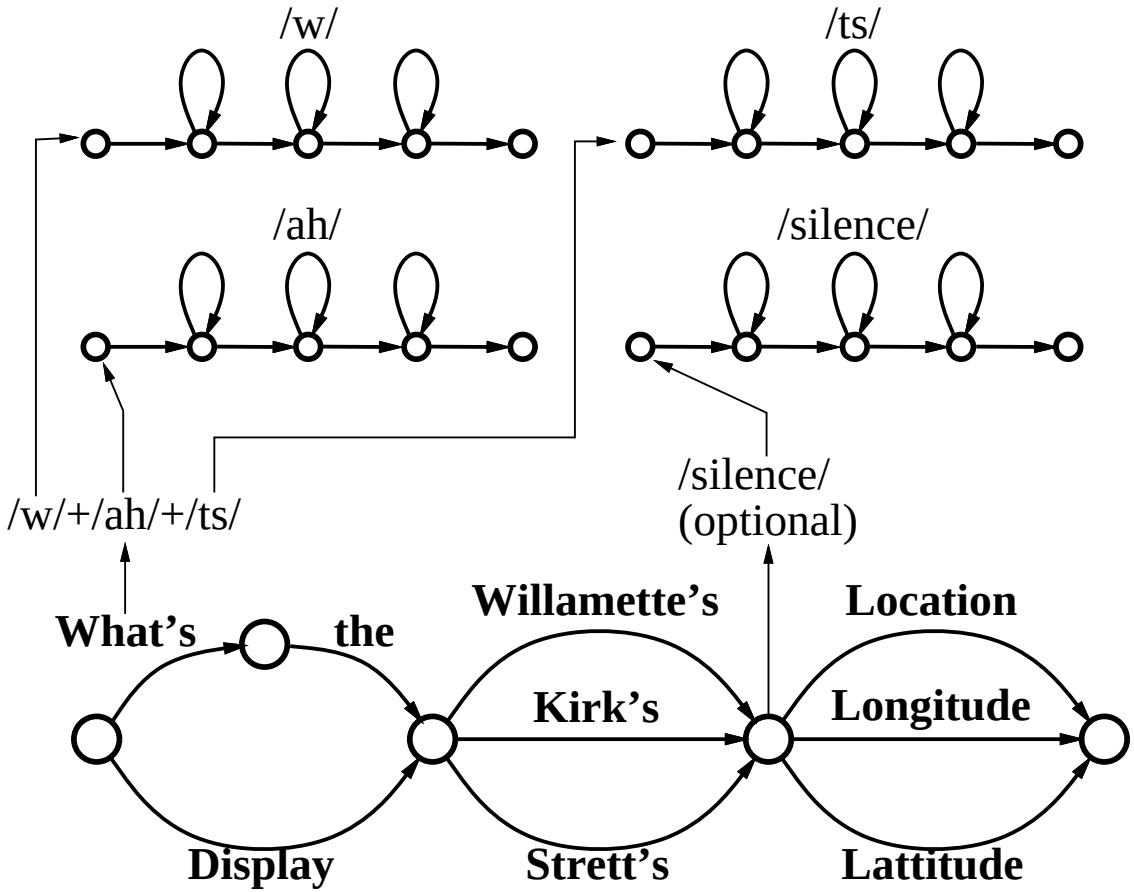
音声パターンYを観測 →発声した内容(音素列)がWである確率(事後確率):

$$P(W|Y) = \frac{P(Y|W)P(W)}{P(Y)}$$

音声認識: $P(Y|W)P(W)$が最大となる音素列仮説Wを求める

- $P(W)$: 音素列Wが発話される確率 → 言語モデル
- $P(Y|W)$: 音素列Wを仮説したとき音声信号Yが観測される条件つき確率(事前確率) → 音素モデル

有限状態オートマトン (FSA)



An HMM network for continuous word recognition with a finite state grammar

図2. 単語をアークに持つような有限状態オートマトン (Sphinx システムの例)

句構造文法

句構造文法 $G = (V_N, V_T, P, S)$
(V_N**: 非終端記号の集合，**V_T**: 終端記号の集合，**P**: 書き換え規則の集合，**S**: 文開始記号)**

書き換え規則: $P = \{\alpha \to \beta \mid \alpha \in V^+, \beta \in V^*, V = V_N \cup N_T\}$

- $|\beta| \geq |\alpha|$**: 文脈依存文法 (Context-sensitive Grammar; CSG)**
- $\alpha \in V_N$**: 文脈自由文法 (Context-free Grammar; CFG)**
- $\alpha \in V_N, \beta \in V_T \cup V_T V_N$**: 正規文法(Regular Grammar; RG)**

確率句構造文法

- **確率的書き換え規則:** $\alpha \xrightarrow{f_{ij}} \beta$
- **確率推定: Inside-Outside アルゴリズム**

文脈自由文法(CFG)による言語モデル

構文解析法 (Parser)

- Earley アルゴリズム
- CYK (Cocke-Younger-Kasami) アルゴリズム
- 一般化 LR 構文解析

その他の文法

- 係受け文法
- 格文法

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

HMM-LR連続音声認識方式 (1)

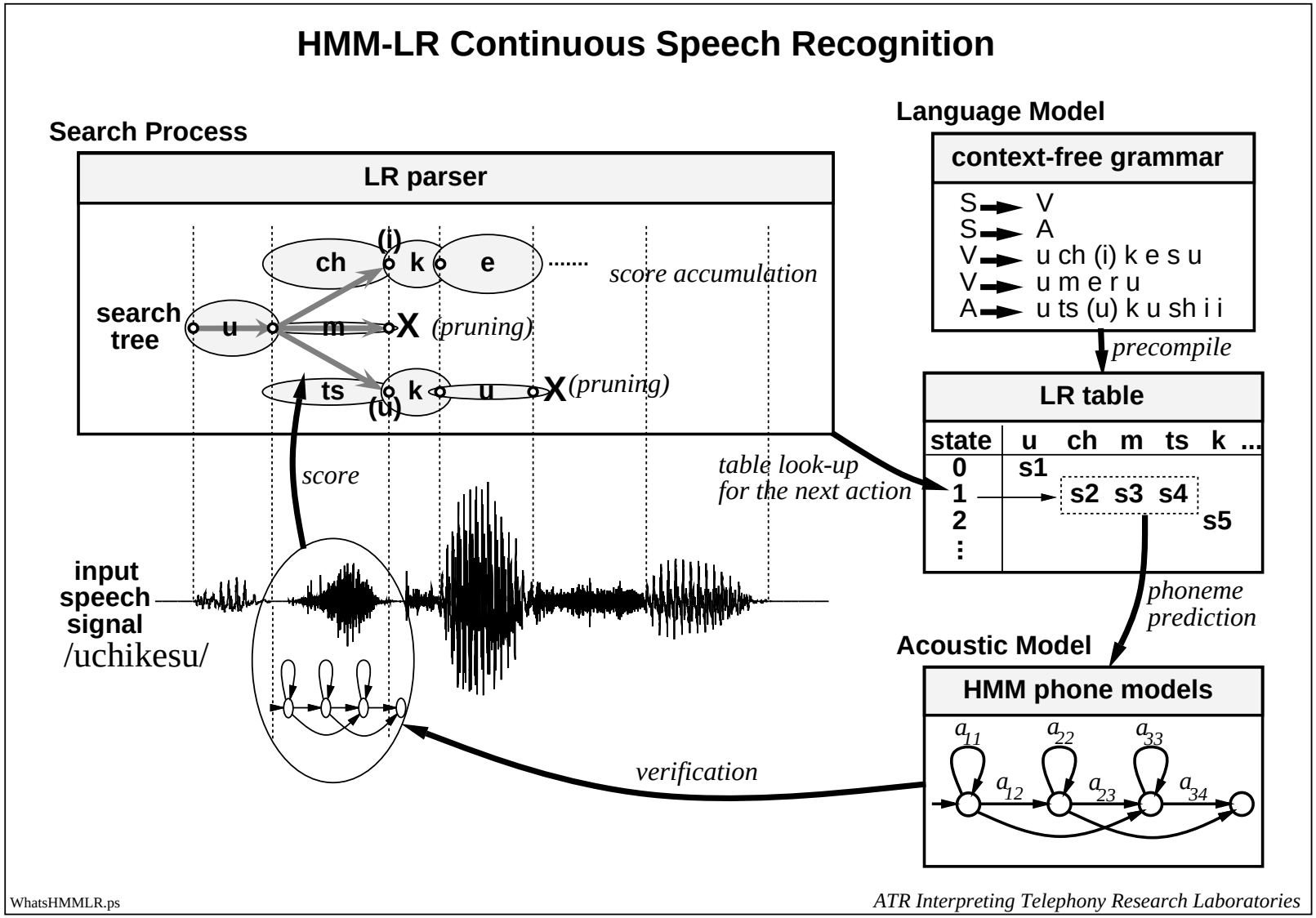


図3. HMM-LR連続音声認識方式 (ATR 1991頃)

HMM-LR連続音声認識方式 (2)

1. 文法の定義

図の文法規則の例は、「打ち消す」、「埋める」、「美しい」という三つの語のみを許す極めて簡単なもので、図のように「書き換え規則」と呼ばれる規則群で表現される。これが、「文脈自由文法」と呼ばれるタイプの文法の記述法である。

2. LRテーブル(表)への変換

文法規則は、事前にLRテーブルと呼ばれる表形式に変換して用いる。図の例では、最初の音素 /u/ が共通なので、入力音声と照合する操作は、最初の音素については１度だけ行なえば良いことになる。このような効率的な動作を規定するのが、LR テーブル (表) である。

3. LRテーブルの構造

このLRテーブルは、状態と呼ばれる番号と、各状態において予測される音素、及び次にどこの状態へ行けばよいかを指示する動作項から構成されている。図では、s1、s2 といった記号が動作項を表し、s1 は音素/u/を予測して状態１へ進むという意味で、状態２では、/ch/、/m/、/ts/の３通りの進み方が書かれている。

4. LR構文解析による音素予測・照合

発話内容は、文法規則に合致しているものとする。例として、HMM-LR が状態０から出発して音素/u/と入力音声とを照合した後、現在の状態が１になったとしよう。このとき、入力音声の認識の途中結果は「u」となっている。LR テーブルで状態１を見ると、この後に予測される音素は、/ch/、/ts/、/r/の３つだけである。HMM-LR は、これら予測された音素にそれぞれ対応する HMM 音素モデルを使用して入力音声を照合し、入力された音声中にこれらの音素が存在する確率を計算し、認識スコア (照合得点) を算出する。図では、音素の認識スコアが/ch/、/ts/、/r/の順に高いことを示している。

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

HMM-LR連続音声認識方式 (3)

5. 枝刈り

次に、HMM-LR は解析済みの音素列「m o」に、新たに予測された/ch/、/ts/、/r/の３つの音素を結合して、「m o ch」、「m o ts」、「m o r」の仮説 (「枝」と言う) を生成する。このとき、計算量の削減のために、認識スコアが低い仮説は今後解析していっても価値がないとみなされて、捨てられる。この処理を枝刈りという。

6. 認識結果の決定

枝刈りを行なった後、再び HMM-LR は LR テーブルを参照して、仮説を進めて構文解析を行なう。このような音素予測照合と枝刈りの動作を繰り返して、文の終端 (LR テーブル中には終端の記号がある) へ到達するまで、複数の仮説 (枝) を生み出したり捨てたりして延ばしていく。終端に到達した枝のうち、認識スコアが最も高いものから順番に第一候補、第二候補、などとして、複数の認識結果を出力する。

以上のような動作をまとめると、「文脈自由文法による文法記述を用い、LR構文解析法により音素予測を行ない、HMM音素モデルを照合する、連続音声認識方式」と言える。このようにして、HMM-LR方式は音声言語処理と音響処理と同時に進行させて無駄のない解析を精度良く進めて、複数の認識候補を得ることができる。

言語モデルの複雑さの尺度

言語Lが有限状態オートマトン(正規文法)で表されているとき，状態jの出現確率$\pi(j)$，状態jから遷移できる単語数$n(j)$を用いて

- **静的分岐数 (static branching factor):** $F_s(L) = \frac{\sum_j n(j)}{\sum_j 1}$
- **平均ファンアウト数 (fanout):** $F_s(L) = \sum_j \pi(j) n(j)$
- **エントロピー (entropy):** $H(L)$

  $H_0(L) = -\sum_w P(w_1, w_2, \cdots, w_k) \log_2 P(w_1, w_2, \cdots, w_k)$

  状態jでの1単語あたりのエントロピー: $H(w|j) = -\sum_w P(w|j) \log_2 P(w|$

  言語のエントロピー: $H(L) = -\sum_j \pi(j) H(w|j)$
- **複雑度 (perplexity):** $F_p(L) = 2^{H(L)}$

  例: Harpy (1000語) $F_p(L) = 4.5$**；IBM レーザ特許文 (1000語)** $F_p(L) = 24$**；10数字連続音声認識** $F_p(L) = 10$
- **音素 perplexity, 等価探索空間** $= 2^{H_0(L)}$

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

連鎖統計による言語モデル

- **単語連鎖確率:** $P(\boldsymbol{W}) = \prod_{i=1}^{n} P(w_i|w_1, w_2, \cdots, w_{i-1})$
- **bigram 統計:** $P(\boldsymbol{W}) = \prod_{i=1}^{n} P(w_i|w_{i-1})$ **(1次マルコフモデル)**
- **trigram 統計:** $P(\boldsymbol{W}) = \prod_{i=1}^{n} P(w_i|w_{i-2}, w_{i-1})$ **(2次マルコフモデル)**
- **Back-off smoothing [Katz]**

構文制御： 音素環境依存 一般化LR構文解析法

一般化LR構文解析による連続音声認識

- 文脈自由文法(CFG)で文法記述 → 高い表現性
- あらかじめコンパイルしてLR表に展開 → 高い効率
- 複数候補が効率良く得られる → 多い情報量

音素環境依存 一般化LR構文解析法 (SSS-LR)

- 隠れマルコフ網(HMnet)を駆動するLRパーザ → 高精度
- 先行音素・当該音素・後続音素の3連鎖をダイナミックに仮説

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

探索手法

対象

- Tree search (木探索)
- Network search (網探索)

手法

- 深さ優先探索 (depth-first search)
  仮説を深さ方向に展開し、バックトラック。つねに仮説は1つ。最も簡単。再帰呼び出しなどで実現。最適解。
- 広がり優先探索 (breadth-frist search)
  深さごとに仮説を全展開。メモリ爆発。最適解。
- 最良優先探索 (best-frist search)
  最もスコアのよいノードから展開。最適解が保証される。
- ビーム探索
  深さごとに仮説を全展開し、上位 B 個(あるいは幅 B)のみ残す。現実的。
- A* 探索
  A 探索では、スコア $f = g + h$ の最大経路を求めるため、ヒューリスティック関数 h の推定値 $\hat{h}$ を用いる。$\hat{h} \leq h$ ならば最適解が得られることが保証される。

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

A アルゴリズム

分岐限定法

分岐限定法は、バックトラックによる枝刈り法の一種。これまでに得られた最善の値と見積もり値の比較によって、不必要な経路探索を省略する。

最短経路問題

各道に重みが与えられているとき、重みの和最小のパスを発見する。

n いま探索を行なっているノード

$\hat{g}(N)$ 開始点から n までの重みの和

A アルゴリズム

1. OPENノードからノード(N_0)をひとつ取り出す。
2. N_0 から遷移できるノード (N_i) を求め。N_0 をCLOSEDにいれる。
3. N_i の内、すでにCLOSEDに入っているものは無視する。
4. 各 N_i について $\hat{g}(N_0, N_i) = \hat{g}(N_0) + c(N_0, N_i)$ を計算し、OPENの中になければ追加する。
5. 既 に OPEN の 中 に あって 、$\hat{g}(N_0, N_i) < \hat{g}(N_i)$ であれば、$\hat{g}(N_i) := \hat{g}(N_0, N_i)$ とし、N_i の親ノードを示すポイン タを N_0 に付け替える。
6. OPENの中を小さい順に並べ直し、$\hat{g}(N)$ 最小のものがスタックトッ プに来るようにする。

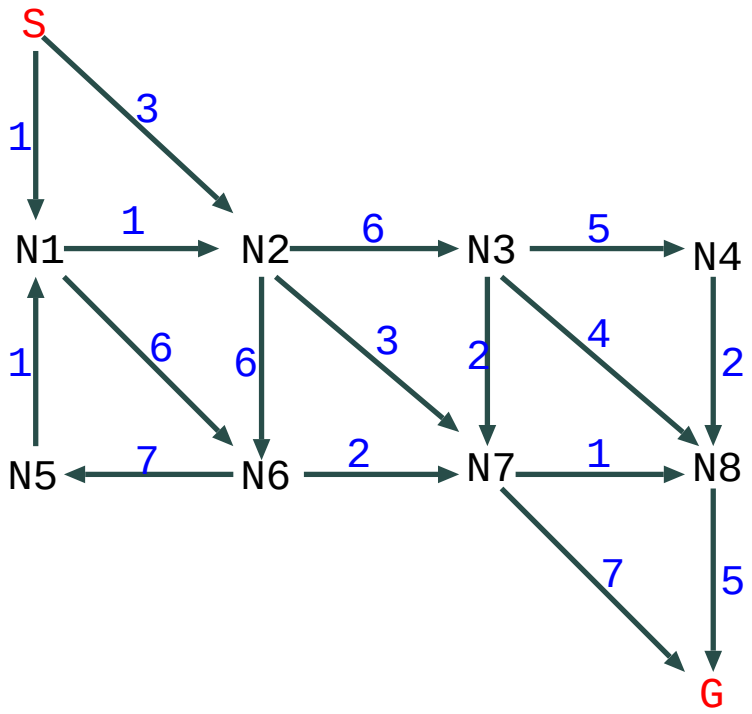


図4. 分岐限定法

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

A アルゴリズム

$OPEN$	$CLOSED$
S	
N_1, N_2	S
N_2, N_6	S, N_1
N_7, N_6, N_3	S, N_1, N_2
N_8, N_6, N_3, G	S, N_1, N_2, N_7
N_6, N_3, G	S, N_1, N_2, N_7, N_8
N_3, G, N_5	$S, N_1, N_2, N_7, N_8, N_6$
G, N_4, N_5	$S, N_1, N_2, N_7, N_8, N_6, N_3$

$$S \rightarrow N_1 \rightarrow N_2 \rightarrow N_7 \rightarrow N_8 \rightarrow G$$

$S \rightarrow N_2$**の代わりに** $S \rightarrow N_1 \rightarrow N_2$ **を生成**

$N_7 \rightarrow G$**の代わりに** $N_7 \rightarrow N_8 \rightarrow G$ **を生成**

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

A アルゴリズム

ヒルクライミング法

ヒューリスティック関数($\hat{h}(N)$)(終了状態までの評価値)を用い、次の１ステップで行ける評価値が最小のところに向かう。
この方法は問題によって局所最適解で止ってしまう可能性がある。
( )内が$\hat{h}(N_i)$の値

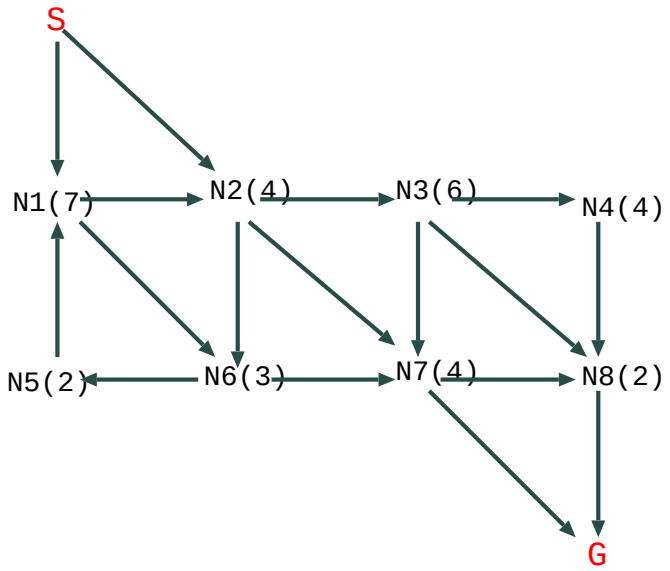


図5. ヒルクライミング法(局所解に陥いる例)

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

Aアルゴリズム

Aアルゴリズム

Aアルゴリズム ＝ 分岐限定法 ＋ ヒルクライミング法

$$\hat{f}(N) = \hat{g}(N) + \hat{h}(N)$$

分岐限定法において一度訪ねたノードでも $\hat{f}$ が小さくなるようなら「過去既に探索開始ノードになった」というマークをはずす。

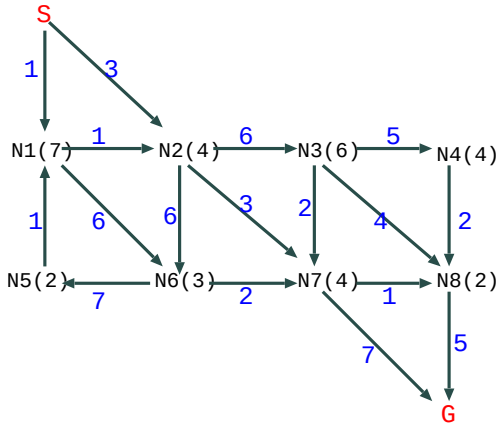


図6. Aアルゴリズム

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

Aアルゴリズム

1. OPENの先頭ノード(N_0)をCLOSEDに入れ、そのすべての子ノード(N_i)について、次式を計算する。

$$\hat{f}(N_0, N_i) = \hat{g}(N_i) + \hat{f}(N_i)$$

1-1. N_iがOPEN,CLOSEDにも含まれないとき、OPENにいれ$\hat{f}(N_i) = \hat{f}(N_0, N_i)$とする。$N_0$から$N_i$へ向かうポインタをつける。

1-2. 1-1.でないとき、$\hat{f}(N_i) > \hat{f}(N_0, N_i)$であれば、$\hat{f}(N_i) := \hat{f}(N_0, N_i)$とし$N_i$の親ノードへのポインタを$N_0$に付け替える。$N_i$が既にCLOSEDに含まれていたときは、$N_i$をOPENに戻す。

2. OPENの中を$\hat{f}$が小さい順に並べ直す。

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

A アルゴリズム

$OPEN$	$CLOSED$
S	
N_2, N_1	S
N_1, N_7, N_6, N_3	S, N_2
N_2, N_7, N_6, N_3	S, N_1
N_7, N_6, N_3	S, N_1, N_2
N_8, N_6, G, N_3	S, N_1, N_2, N_7
N_6, G, N_3	S, N_1, N_2, N_7, N_8
G, N_3, N_5	$S, N_1, N_2, N_7, N_8, N_6$

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

A アルゴリズム

1. $S \rightarrow \{\underline{N_2(7)}, N_1(8)\}$
2. $S \rightarrow N_2 \rightarrow \{\underline{N_1(8)}, N_3(15), N_6(12), N_7(10)\}$
3. $S \rightarrow N_1 \rightarrow \{\underline{N_2(6)}, N_6(10), N_3(15), N_7(10)\}$
4. $S \rightarrow N_1 \rightarrow N_2 \rightarrow \{N_3(14), \underline{N_7(9)}, N_6(10)\}$
5. $S \rightarrow N_1 \rightarrow N_2 \rightarrow N_7 \rightarrow \{\underline{N_8(8)}, G(12), N_3(14), N_6(10)\}$
6. $S \rightarrow N_1 \rightarrow N_2 \rightarrow N_7 \rightarrow N_8 \rightarrow \{G(11), N_3(14), \underline{N_6(10)}\}$
7. $S \rightarrow N_1 \rightarrow N_6 \rightarrow \{\underline{G(11)}, N_3(14), N_5(16)\}$
8. $S \rightarrow N_1 \rightarrow N_2 \rightarrow N_7 \rightarrow N_8 \rightarrow \{\underline{G(11)}, N_3(14), N_5(15)\}$

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

Aアルゴリズム

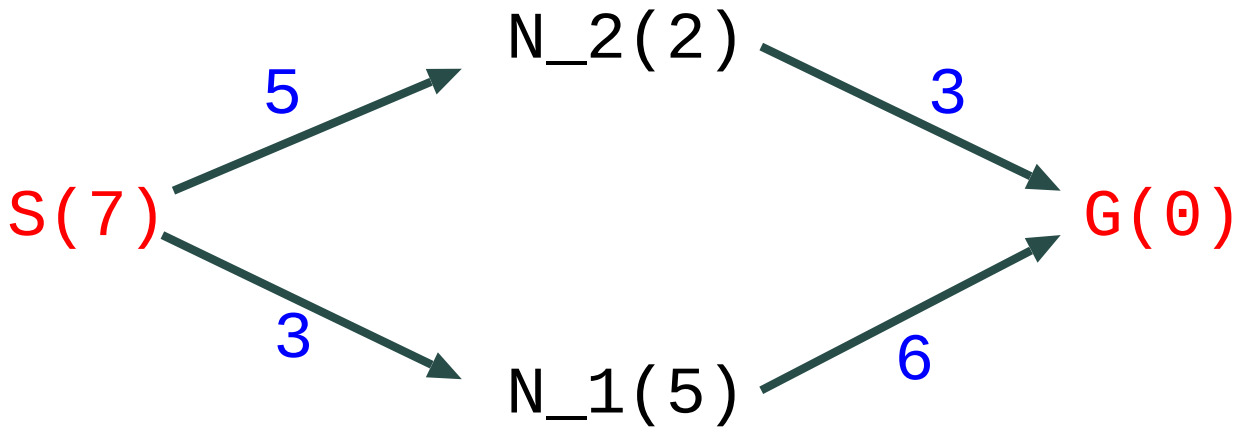


図7. Aアルゴリズムがうまくいく例

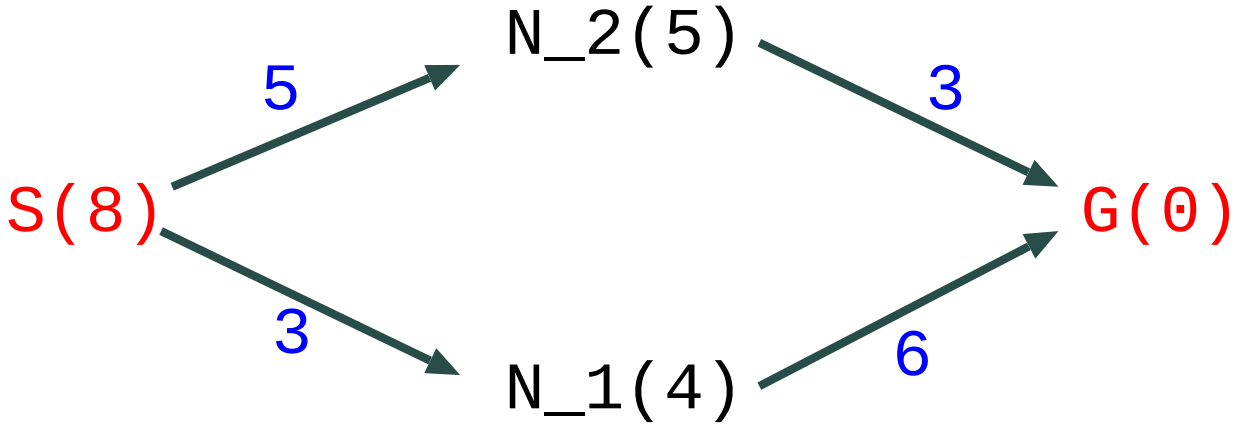


図8. Aアルゴリズムが失敗する例

Aアルゴリズムは必ず探索に成功するが、その経路が最適解とは限らない。

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

A^* アルゴリズム

A^* アルゴリズム

A アルゴリズムは必ず探索に成功するが、$\hat{h}(N) > h(N)$ だと $\hat{f}(N)$ が過大に評価され最適順路を見つけられる保証はない。
すべてのノードにおいて $\hat{h}(N) \leq h(N)$ が保証され必ず最適解が見つけられる場合を A^* アルゴリズムという。

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

N-bestアルゴリズム

N-bestアルゴリズム

N-bestアルゴリズムは、Viterbiアルゴリズムを改良したアルゴリズムであり、最も確率の高い状態系列及びそのパスの確率から、N番目に確率の高い状態系列及びそのパスの確率までを求めることができる。

N-bestパス探索

今までのViterbiアルゴリズムでは、尤度の高い状態系列を１本だけ求めていた。N-best探索では、尤度の高い方からN個の状態系列を求める。

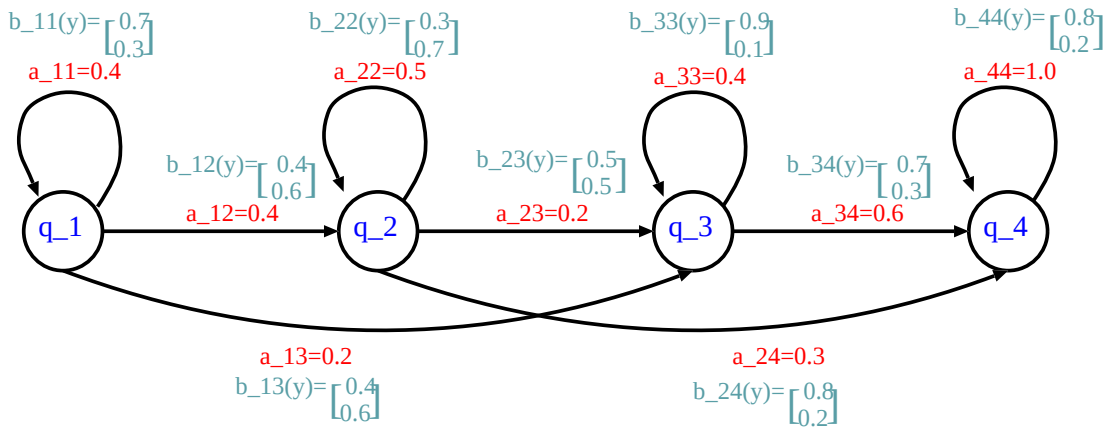


図9. HMMを用いた例(1-best探索)

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

N-bestアルゴリズム

Viterbiアルゴリズムでいったん最後まで解析を済ませれば、A^*アルゴリズムを用いて、n-bestのパスを求めることができる。

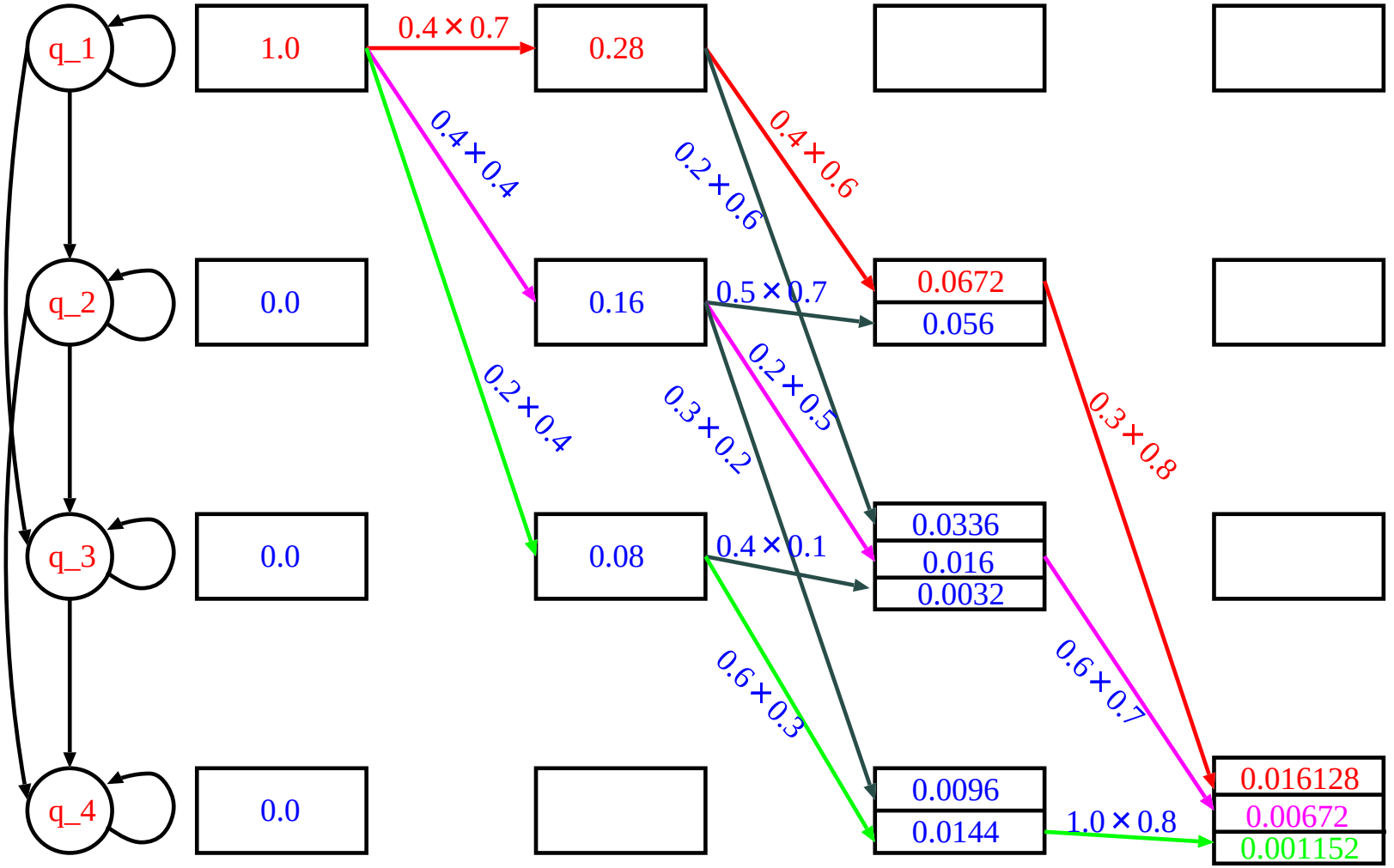


図10. HMMを用いた例(3-best探索)

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

スタックデコーダ

与えられた入力音声の特徴パラメータ時系列y_1^nに対して、最適な単語列wを求める問題を考える。
しかし、異なるフレーム間での尤度を単純に比較することはできない。

そこで、A* アルゴリズムを用い、音声区間全体の推定尤度を求めて比較する方法がある。これは、Viterbiアルゴリズムとは異なり、IBMのグループによってスタックデコーディングアルゴリズムとして、音声認識に初めて適用された。

基本的には単語列wの尤度の$\Lambda(w)$の大きい順に単語列を生成（展開）させていく。長さの異なる単語列を比較する為に尤度$\Lambda(w)$の設定法が重要となる。

注意：時系列y_1^nとは、$\{y_m\}_{m=1}^n$の意味である。

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

スタックデコーディングアルゴリズム

単語列wが、音声入力の特徴パラメータ時系列y_1^nの先頭からある時点までに発声されたとみなされる確率は、

$$\sum_{i=1}^{n} P(\boldsymbol{w}, \boldsymbol{y}_1^i) = P(\boldsymbol{w}) \sum_{i=1}^{n} P(\boldsymbol{y}_1^i | \boldsymbol{w})$$

右辺第１項は言語の確率モデルから事前に計算できるが、単語長と共に減少するので、異なる単語列同士の比較はできない。そこで、単語列長による正規化が必要となり、最適な単語列に対しては単語長と共に増加し、その他の単語列に対しては減少する次の評価関数を用いる。

$$\Lambda(\boldsymbol{w}) = \sum_{i=1}^{n} P(\boldsymbol{w}, \boldsymbol{y}_1^i) \alpha^{n-i} \sum_{\boldsymbol{w}'} P(\boldsymbol{w}', \boldsymbol{y}_{i+1}^n | \boldsymbol{w}, \boldsymbol{y}_1^i)$$

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

スタックデコーディングアルゴリズム

$$\Lambda(\boldsymbol{w}) = \sum_{i=1}^{n} P(\boldsymbol{w}, \boldsymbol{y}_1^i)\alpha^{n-i} \sum_{\boldsymbol{w}'} P(\boldsymbol{w}', \boldsymbol{y}_{i+1}^n | \boldsymbol{w}, \boldsymbol{y}_1^i)$$

上式の$\Sigma_{\boldsymbol{w}'} P(\boldsymbol{w}', \boldsymbol{y}_{i+1}^n | \boldsymbol{w}, \boldsymbol{y}_1^i)$は、$w$の対応する入力音声区間$y_1^i$の残りの音声区間$y_{i+1}^n$で$w'$が$w$に連接する期待値を示す。ここで、正規化定数$\alpha$は、$\Lambda(w)$の増加傾向を制御するもので実験的に決められる。
つまり、

$$\alpha^{n-i} \sum_{\boldsymbol{w}'} P(\boldsymbol{w}', \boldsymbol{y}_{i+1}^n | \boldsymbol{w}, \boldsymbol{y}_1^i)$$

はA*アルゴリズムのヒューリスティック関数に対応している。

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

スタックデコーディングアルゴリズム

実際には、

$$\sum_{\boldsymbol{w}'} P(\boldsymbol{w}', \boldsymbol{y}_{i+1}^{n} | \boldsymbol{w}, \boldsymbol{y}_{1}^{i})$$

を求めることはできないので、wに独立であるとして、更に$P(\boldsymbol{w}', \boldsymbol{y}_{i+1}^{n} | \boldsymbol{y}_{1}^{i})$の平均値$E(\boldsymbol{y}_{i+1}^{n} | \boldsymbol{y}_{1}^{i})$ をマルコフ過程による近似から訓練データによって求められる。すなわち、

$$E(\boldsymbol{y}_{i+1}^{n} | \boldsymbol{y}_{1}^{i}) = \prod_{j=i+1}^{n} E(y_i | \boldsymbol{y}_{i-k}^{j-1}) = \prod_{j=i+1}^{n} E(y_i | y_{i-1})$$

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

スタックデコーディングアルゴリズム

これにより、不完全なパス(まだ完全な文でない単語列、照合されていない入力音声区間が存在する場合)に対しては、

$$\Lambda(\boldsymbol{w}) = P(\boldsymbol{w}) \sum_{i=1}^{n} P(\boldsymbol{y}_1^i|\boldsymbol{w})\alpha^{n-i}E(\boldsymbol{y}_{i+1}^n|\boldsymbol{y}_1^i)$$

また、完全なパス(全入力音声が照合された場合)に対しては、

$$\Lambda(\boldsymbol{w}) = P(\boldsymbol{w})P(\boldsymbol{y}_1^n|\boldsymbol{w})$$

$\boldsymbol{w} = \boldsymbol{w}_1^k$は$\boldsymbol{w}_1^{k-1}w_k$と表現できるから、任意の$i \leq n$に対して、$P(\boldsymbol{y}_1^i|\boldsymbol{w}_1^{k-1})$の結果が決まっていれば、$P(\boldsymbol{y}_1^i|\boldsymbol{w}_1^k)$、すなわち、$\Lambda(\boldsymbol{w})$が計算できる。

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

スタックデコーディングアルゴリズム

探索手続きは次のようにして行う。

1. スタックに既に評価された単語列wとその尤度$\Lambda(w)$の値を$\Lambda(w)$の大きい順に積む。
2. スタックの一番上にある単語列の$\Lambda(w)$の値と閾値δ以内の差を持つ単語列を展開（単語を接続）し、その結果を$\Lambda(w)$の大きさでソーティングしながらスタックに積む。
3. スタックの先頭にある単語列が完全なパス（$\Lambda(w) = P(w)P(y_1^n|w)$によって求められたもの）であるなら探索を終了し、この単語列を認識結果とする。

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)

参考文献

■ 中川 聖一「確率モデルによる音声認識」コロナ社

嵯峨山茂樹 (東京大学大学院情報理工学系研究科システム情報学専攻)