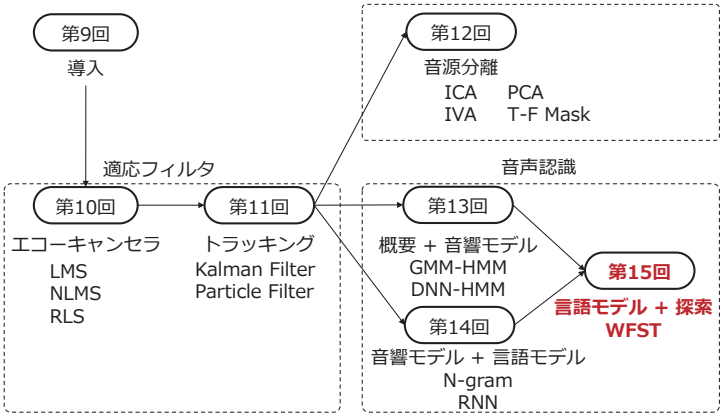
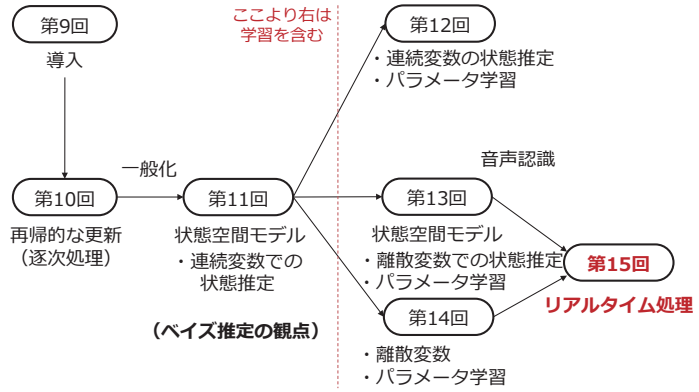
知的情報処理論 第15回

2023年7月25日（火）
武田

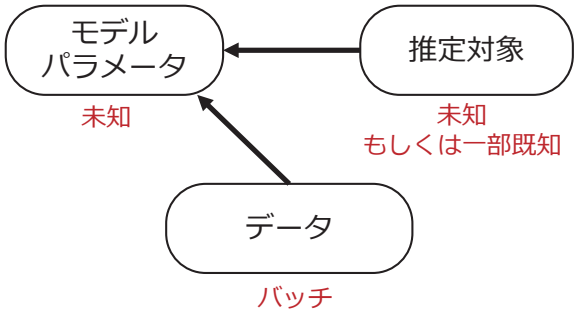
各回の内容（予定）



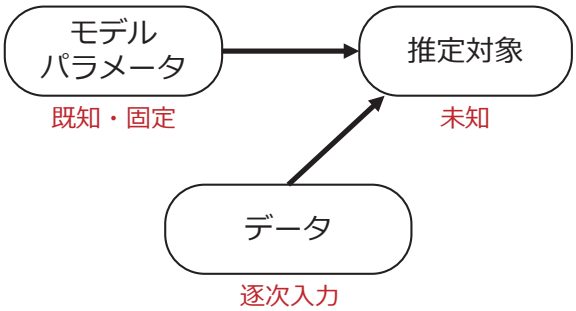
各回の内容（予定）



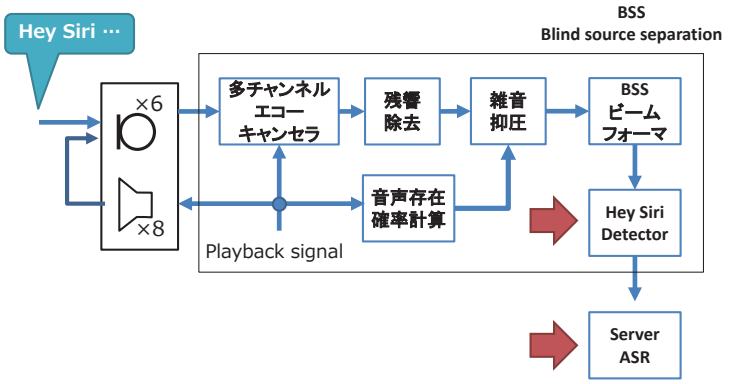
今回の話題: 学習



今回の話題: 認識



Apple Home Pod の構成 (SLT2018より)



本日の内容: 音声認識

1. 言語モデル II （オマケ）
   - Neural 言語モデル
     （前半パートでもあったはずなので）
2. 探索 (メイン)
   - ビタビアルゴリズム
   - ビームサーチ
   - WFST
3. End-to-End 音声認識 （オマケ）

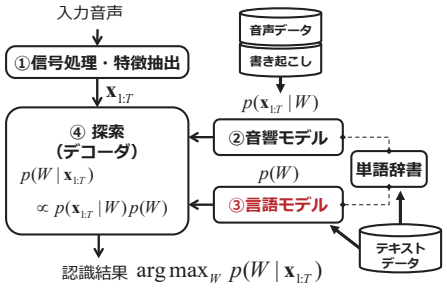
音声認識

音声信号をテキストに変換

音声認識

音響モデル: (Infinite) GMM

言語モデル: 教師なし単語分割



③ 言語モデル II

基本的な NEURAL 言語モデル

ニューラル言語モデル

言語モデルでやりたいこと

$p(\mathbf{w}_n, \mathbf{w}_{n-1}, \mathbf{w}_{n-2}, \dots)$
$p(\mathbf{w}_n | \mathbf{w}_{n-1}, \mathbf{w}_{n-2}, \dots)$ のモデル化

自然言語などの記号の扱い ☹ 難しい

– 背後に法則や原理があるのか/ないのか cf. 物理
– 履歴長・スパースネス問題 (古典 N-gram)

ニューラルネットワークの応用

– 大量のテキストデータを活用しとにかく学習

※ 以降に登場するネットワーク構造自体は言語モデルだけに適用されるものではない

補足: スコア計算と単語列生成

スコア計算

– 対象の単語列 $\mathbf{w}_n, \mathbf{w}_{n-1}, \dots, \mathbf{w}_1$ は **given**
– $p(\mathbf{w}_1)$, $p(\mathbf{w}_2|\mathbf{w}_1)$, $\cdots$ $p(\mathbf{w}_n|\mathbf{w}_{n-1}, \mathbf{w}_{n-2}, \dots)$ **の値**をモデルに従って計算

用途: 候補(仮説)集合からモデルに適合する文を選択

単語列生成

– サンプリング・探索のような手続き

例1. (前から) 順番に draw: $\tilde{\mathbf{w}}_1 \sim p(\mathbf{w}_1)$, $\tilde{\mathbf{w}}_2 \sim p(\mathbf{w}_2|\tilde{\mathbf{w}}_1)$, $\cdots$, $\tilde{\mathbf{w}}_n \sim p(\mathbf{w}_n|\tilde{\mathbf{w}}_{n-1}, \tilde{\mathbf{w}}_{n-2}, \dots \tilde{\mathbf{w}}_1)$

例2. 高いスコアの系列を（スコア計算を行いながら）優先的に生成（探索）cf. パーティクルフィルタ

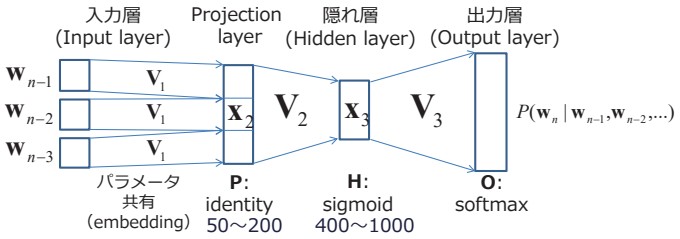
フィードフォワード型

モデル: $P(\mathbf{w}_n | \mathbf{w}_{n-1}, \mathbf{w}_{n-2}, \dots)$

入力: 過去 n-1 単語

表現 - one-hot vector $\mathbf{w}_n = [0,\dots,0,1,0,\dots,0]$

語彙サイズ次元V（40k～200k）
その単語の "ID" に該当する次元部分のみ 1



フィードフォワード型

補足: one-hot vector = 列ベクトルの抽出

$\mathbf{w}_n = [0,\dots,0,1,0,\dots,0]$ k 番目の要素のみ 1 次元: V

$\mathbf{V}_1$ 次元: L × V

Embedding!!!

$$\mathbf{V}_1\mathbf{w}_{n-1} = \begin{bmatrix} v_{11} & \cdots & v_{1(k-1)} & v_{1k} & v_{1(k+1)} & \cdots & v_{1V} \\ \vdots & & & \vdots & & & \vdots \\ v_{(i-1)1} & & & v_{(i-1)k} & & & v_{(i-1)V} \\ v_{i1} & \cdots & v_{i(k-1)} & v_{ik} & v_{i(k+1)} & \cdots & v_{iV} \\ v_{(i+1)1} & & & v_{(i+1)k} & & & v_{(i+1)V} \\ \vdots & & & \vdots & & & \vdots \\ v_{I1} & \cdots & v_{L(k-1)} & v_{Lk} & v_{L(k+1)} & \cdots & v_{LV} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

フィードフォワード型

基本: 重み付き和＋バイアス

$$\mathbf{x}_{i+1} = \mathbf{V}_i\mathbf{x}_i + \mathbf{b}_i$$

i: 層インデックス
$\mathbf{x}_i$: i 層目への入力
$\mathbf{V}_i, \mathbf{b}_i$: i 層目の重み行列とバイアスベクトル

ノード毎の演算:

出力層: ソフトマックス演算

$$y_j = \frac{\exp(x_j)}{\sum \exp(x_j)}$$

• 多クラスに対する確率を表現

中間層: 非線形変換

例： シグモイド関数

$$y_j = \frac{1}{1+\exp(-x_j)}$$

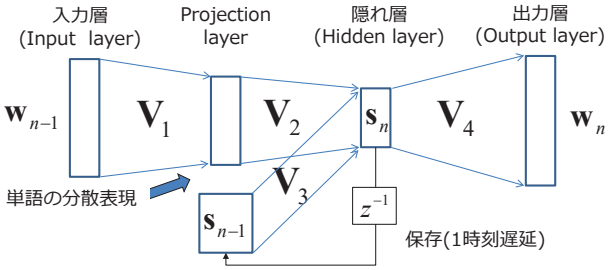
Projection layer: 恒等関数

リカレント型

モデル: $P(\mathbf{w}_n|\mathbf{s}_{n-1}, \mathbf{w}_{n-1})$

– 1つ前の隠れ層の出力をフィードバック

– 仮想的に全ての履歴を考慮 = “状態”

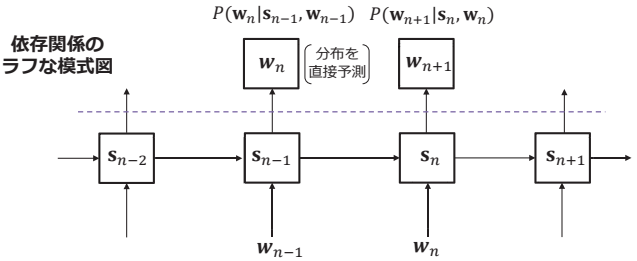


リカレント型

モデル: $P(\mathbf{w}_n|\mathbf{s}_{n-1}, \mathbf{w}_{n-1})$

– 1つ前の隠れ層の出力をフィードバック

– 仮想的に全ての履歴を考慮 = “状態”



$\mathbf{s}_n$ は曖昧性を持たないと仮定＝確定的に決まる（分布は仮定されない）

Encoder-Decoder 型

モデル: $P(\mathbf{w}_m, \dots, \mathbf{w}_1|\boldsymbol{x}_n, \dots, \boldsymbol{x}_1)$

– $\boldsymbol{x}_n, \dots, \boldsymbol{x}_1$: 条件部分の変数

e.g. context の単語列, 音声信号, etc…

– エンコーダ: 特徴表現の生成

$$\boldsymbol{y}_h, \dots, \boldsymbol{y}_1 = \boldsymbol{f}(\boldsymbol{x}_n, \dots, \boldsymbol{x}_1)$$

– デコーダ: 条件付き分布の表現

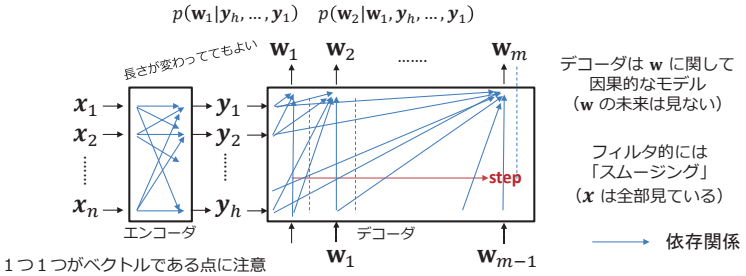
$$P(\mathbf{w}_n, \dots, \mathbf{w}_1|\boldsymbol{y}_h, \dots, \boldsymbol{y}_1)$$
$$= P(\mathbf{w}_1|\boldsymbol{y}_h, \dots, \boldsymbol{y}_1)P(\mathbf{w}_2|\mathbf{w}_1, \boldsymbol{y}_h, \dots, \boldsymbol{y}_1)\cdots$$
$$P(\mathbf{w}_m|\mathbf{w}_{m-1}, \dots, \mathbf{w}_1, \boldsymbol{y}_h, \dots, \boldsymbol{y}_1)$$

Encoder-Decoder 型

モデル: $P(\mathbf{w}_m, \dots, \mathbf{w}_1|\boldsymbol{x}_n, \dots, \boldsymbol{x}_1)$

– エンコーダ: 特徴表現の生成

– デコーダ: 条件付き分布の表現



1つ1つがベクトルである点に注意

系列変換の例: End-to-End なんちゃってカナ漢字変換

T5 small + Tokenizer: Rinna Ja neox 3.6b
音声Corpus書き起こし+Wiki40B: 約1500万文
おおよその文は変換できているが…

注意: 大枠の構造と細かい構造

大枠の構造: 確率モデル・概念レベル

– 自己回帰, 条件付き分布, etc… (グラフィカルモデルレベルなども)

– 正規化, 再帰型, エンコーダ・デコーダ, etc…

→ 詳細は “細かい構造” で具体化

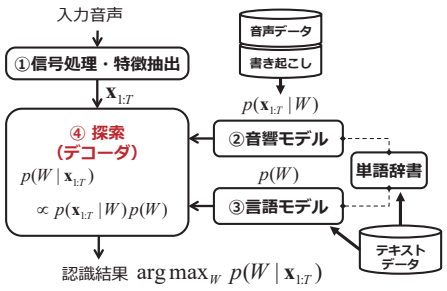
細かい構造: 具体化したネットワーク

– 非線形関数, LSTM, attention, etc..

• LSTM, transformer などは中規模ぐらいか？

– layer norm., positional encoding, etc…

細かいレベル: 次から次に新しいものが出現 ☹
大枠レベル: 基本的なモデルや概念に基づいたり問題に合わせて設計



④ 探索（デコーダ）

モデルは構築済み=パラメータは学習済み

リアルタイム音声認識

元々の定式化

$$\hat{W} = \arg\max_W p(\mathbf{x}_{1:T} \mid W)p(W)$$
$$= \arg\max_W \underbrace{p(W)}_{\text{言語モデル}} \sum_{s_{1:T}} \underbrace{p(\mathbf{x}_{1:T} \mid s_{1:T}, W)p(s_{1:T} \mid W)}_{\text{音響モデル（HMM）}}$$

☹ **文毎の全状態を列挙＆スコア計算は非現実的**

→ 何かしらの効率化が不可欠

① スコア関数の近似: ビタビサーチ
② 探索の工夫: ビームサーチ（枝刈り）
③ 探索（状態）空間のコンパクト化: WFST

音声認識に限らず系列データ処理一般に使われる

① スコア関数の近似

再定式化: 最尤な「系列」の探索

– 興味のあること: 尤もらしい音素列，単語列

☺ 尤もスコアが高い系列（パス）の計算

sum → max 演算

$$\hat{W} = \arg\max_W p(W) \sum_{s_{1:T}} p(\mathbf{x}_{1:T} \mid s_{1:T}, W)p(s_{1:T} \mid W)$$

ビームサーチ（枝刈り）と組み合わせて計算省略

$$\hat{W} = \arg\max_W p(W) \max_{s_{1:T}} p(\mathbf{x}_{1:T} \mid s_{1:T}, W)p(s_{1:T} \mid W)$$

ビタビアルゴリズムで効率的に計算
Max-sum algorithm

HMM の最尤パスの計算(1)

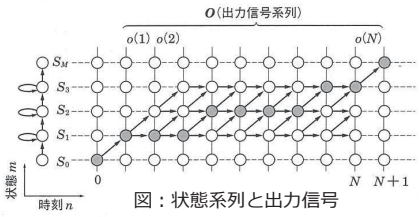
入力: 信号/特徴量の系列 $\mathbf{x}_{1:T}$

出力: 最も高い確率で出力する状態遷移系列(パス)

(HMM: 同じ出力を行う経路が複数存在する)

$$\hat{s}_{1:T} = \arg\max_{s_{1:T}} p(\mathbf{x}_{1:T} \mid s_{1:T})p(s_{1:T})$$
$$= \arg\max_{s_{1:T}} [\ln p(\mathbf{x}_{1:T} \mid s_{1:T}) + \ln p(s_{1:T})]$$

W: 省略



図：状態系列と出力信号

HMM の最尤パスの計算(2)

ビタビアルゴリズム (Viterbi algorithm)

– ある状態に至る最も確からしい系列を算出

– 時刻 t に関して再帰的にスコア計算が可能

$$\max_{s_{1:t}} \ln p(\mathbf{x}_{1:t}, s_{1:t}) = \max_{s_{1:t}} [\ln p(\mathbf{x}_{1:t} \mid s_{1:t}) + \ln p(s_{1:t})]$$
$$= \max_{s_{1:t}} [\sum_{k=1}^{t} [\ln p(\mathbf{x}_k \mid s_k) + \ln p(s_k \mid s_{k-1})]]$$
$$= \max_{s_t} [\ln p(\mathbf{x}_t \mid s_t) + \max_{s_{1:t-1}} \{\ln p(s_t \mid s_{t-1}) + \sum_{k=1}^{t-1} \ln p(\mathbf{x}_k \mid s_k)p(s_k \mid s_{k-1})\}]$$
$$= \max_{s_t} [\ln p(\mathbf{x}_t \mid s_t) + \max_{s_{t-1}} \{\ln p(s_t \mid s_{t-1}) + \max_{s_{1:t-2}} \ln p(\mathbf{x}_{1:t-1}, s_{1:t-1})\}]$$

1step データとの適合度　1step 状態遷移コスト

HMM の最尤パスの計算(2)

ビタビアルゴリズム (Viterbi algorithm)

– ある状態に至る最も確からしい系列を算出

– 時刻 t に関して再帰的にスコア計算が可能

$$D(s_t) = \max_{s_{1:t-1}} \ln p(\mathbf{x}_{1:t}, s_{1:t})$$ と定義（状態 s_t の関数）

$$= \ln p(\mathbf{x}_t \mid s_t) + \max_{s_{t-1}} \{\ln p(s_t \mid s_{t-1}) + \max_{s_{1:t-2}} \ln p(\mathbf{x}_{1:t-1}, s_{1:t-1})\}$$
$$= \ln p(\mathbf{x}_t \mid s_t) + \max_{s_{t-1}} \{\ln p(s_t \mid s_{t-1}) + D(s_{t-1})\}$$

初期値 $D(s_1) = \ln p(\mathbf{x}_1 \mid s_1) + \ln(s_1)$

最大値 $\max_{s_T} D(s_T)$

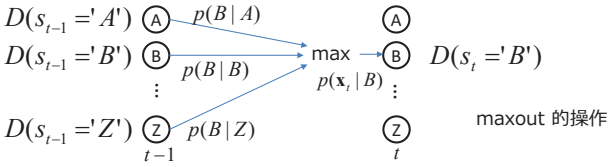
HMM の最尤パスの計算(2)

ビタビアルゴリズム (Viterbi algorithm)

– ある状態に至る最も確からしい系列を算出
– 時刻 t に関して再帰的にスコア計算が可能

初期値 $D(s_1) = \ln p(\mathbf{x}_1 \mid s_1) + \ln(s_1)$

$$D(s_t) = \ln p(\mathbf{x}_t \mid s_t) + \max_{s_{t-1}} \{\ln p(s_t \mid s_{t-1}) + D(s_{t-1})\}$$



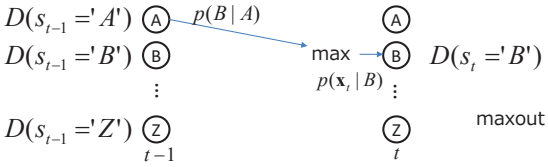
maxout 的操作

HMM の最尤パスの計算(2)

ビタビアルゴリズム (Viterbi algorithm)

– ある状態に至る最も確からしい系列を算出
– 時刻 t に関して再帰的にスコア計算が可能
– **選択されたパスをそれぞれ記録**
→ 最後に辿る（バックトラック）= 最尤パス

$\max_{s_T} D(s_T)$



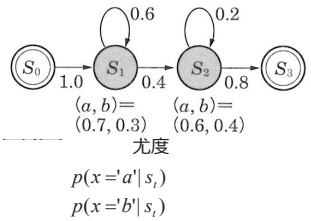
maxout 的操作

HMM の最尤パスの計算(3)

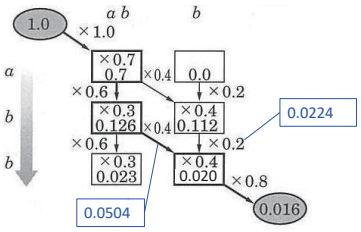
単純な計算例

– 離散的な「記号」が観測される場合
入力: “abb”, 隠れ状態数: 2

状態遷移図

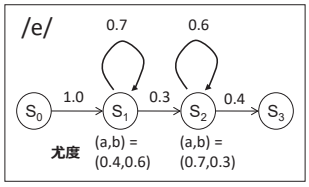


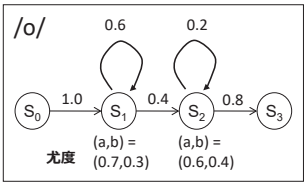
トレリス（縦が時刻）



Mini Quiz #1

- 与えられた観測列 aab （3フレーム）が1音素だとして，これに対する最尤の音素はどちらか
  – 簡単化のために，1フレームあたりの尤度計算の代わりに，観測されたシンボルとその尤度を与えている．本来これらは多次元 GMM などで計算される



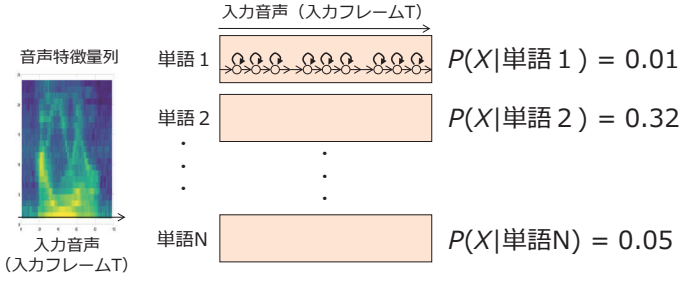


② 探索の工夫

デコーダの役割

入力に対し最も尤もらしい単語系列を得る

– ナイーブな計算法: 単語認識の場合



※単語認識: $P(W)$=等確率（$1/N$）

デコーダの役割

単語認識

例: 語彙サイズ: 20000語

⇒ 20000語に対して P(X|単語) を計算

連続文認識

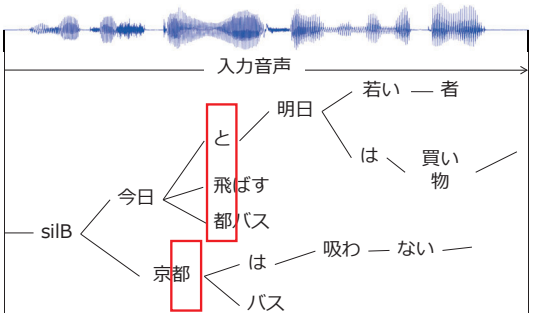
例: 1文が平均10単語からなる場合

全探索: $20000^{10} \fallingdotseq 10^{43}$ 通りの組み合わせ

各時点での候補を10単語に絞っても 10^{10} 通り

リアルタイム処理: 解の探索を動的に制御する必要

膨大な探索空間



- 単語は任意の位置から始まり得る
- 全探索は不可能

デコーダの役割

☹ できるだけ早く解を得たい

- リアルタイム性は重要: not batch
- 「制約を満たす解を一つ求めればよい」問題ではない

→「最も良い（尤もらしい）解を求める問題」

☹ 全探索は不可能

→ 仮説を動的に展開しながら探索を行い，解がある可能性の高い部分だけ尤度を計算

② 探索の工夫: ビームサーチ

処理量を削減

– **最尤系列**以外は余分な計算 ⇒ 削減可能

– ある時点でスコアがかなり低いノードは最終的な最尤パスの一部になる見込みが薄い

⇩ **「ビーム幅」を設定**

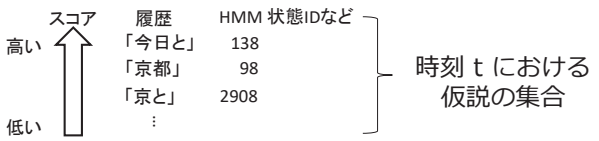
尤度上位から一定数のノードのみ計算
下位のノードを計算から除外（枝刈り）

- 仮説集合（系列群）を作成して管理
- パーティクルフィルタにイメージが近い（非確率的）

② 探索の工夫: ビームサーチ

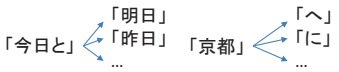
補足: 仮説の展開（フレーム同期の場合）

– 仮説: ある時刻 t における認識結果の候補など

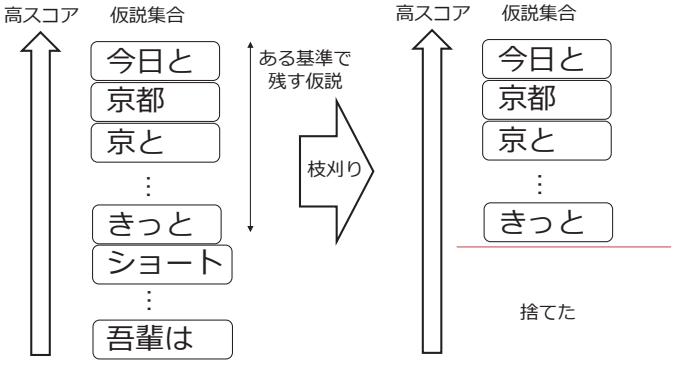


– 仮説の展開: 次の時刻の状態へ遷移（場合分け）

- HMM 状態: 停留 or 次状態
- 後続の単語接続:



② 探索の工夫: ビームサーチ



ビームの性質

狭くするほど

☺ 計算量削減の効果大

☹ 最尤パスが途中で枝刈りの危険性大

広くするほど

☹ 計算量削減の効果小

☺ 最尤パスの枝刈りの危険性小

ビーム幅 ⇔ 認識率： トレードオフ

Julius の認識アルゴリズムの場合 2パス構成

第1パス: フレーム同期ビーム探索

– 粗く探索し，候補単語とその位置を出力
  • 音響モデル: 単語間 tri-phone の計算を簡略化
  • 言語モデル: 単語2-gram

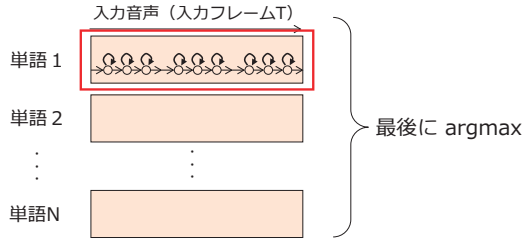
第2パス

– 第1パスの結果をヒューリスティックとした A*探索
– 第1パスよりも精細なモデルで尤度を計算
  • 言語モデル：単語3-gram

第1パス フレーム同期ビーム探索

演算順序が単語単位の場合

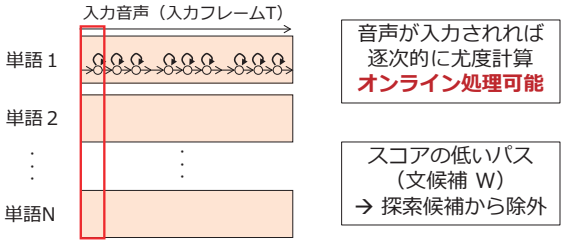
For all w (辞書中の単語)
  For all T (入力フレーム)
    Proceed Viterbi



第1パス フレーム同期ビーム探索

演算順序がフレーム単位の場合

For all T (入力フレーム)
  For all w (辞書中の単語)
    Proceed Viterbi

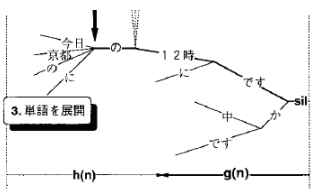


第2パス： 逆向きA*探索

発話の終端から始端に向けて精細に探索

ある種のリスコアリング (スムージング)

第1パスの結果を未探索部分のヒューリスティック（推定値）として、より詳細に計算



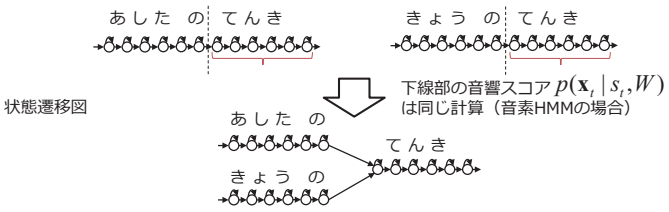
- 第1パスのビーム内に残った単語（とその位置）を利用
- 最良優先探索
- 評価関数
  $f(n) = g(n) + h^*(n)$
  $h^*(n)$: 未展開部の推定スコア
  第1パスでの近似尤度

③ 探索空間のコンパクト化

③ 探索空間のコンパクト化

探索と音響スコア計算を効率化

– 似たような文は共通のHMMを包含
– 候補の文が有限なら共通化可能



探索空間のマージにより音響スコア共有実装も容易

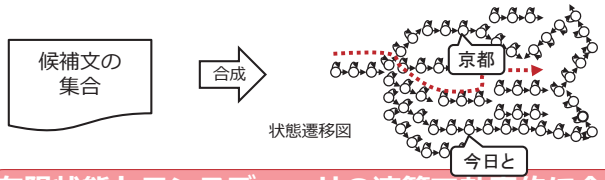
※ 探索の履歴は別々で保持

③ 探索空間のコンパクト化

探索と音響スコア計算を効率化

– 似たような文は共通のHMMを包含
– 候補の文が有限なら共通化可能

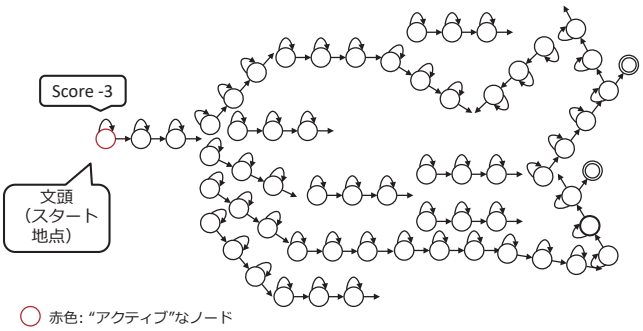
→ **認識＝静的な巨大ネットワーク上のパス探索**



有限状態トランスデューサの演算で効率的に合成
プログラムがシンプルに & 探索漏れも削減

③ 探索空間のコンパクト化

特徴量1フレーム目の入力

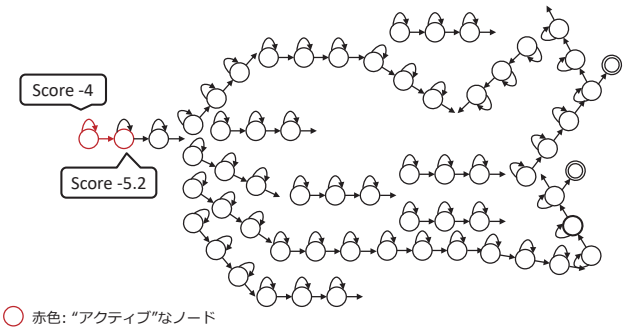


○ 赤色: “アクティブ”なノード

③ 探索空間のコンパクト化

特徴量2フレーム目の入力

ネットワーク上を「分裂する “すごろくのコマ” 」で１つずつ進める．
その際，言語スコア，遷移スコア，尤度スコアを（対数上で）加算していく．

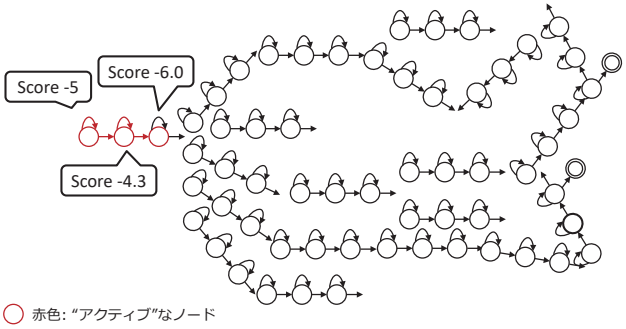


○ 赤色: “アクティブ”なノード

③ 探索空間のコンパクト化

特徴量3フレーム目の入力

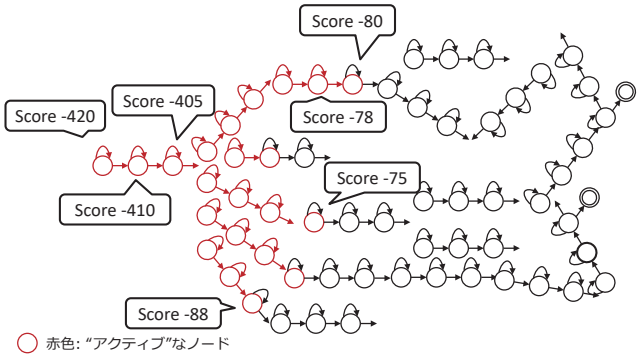
ネットワーク上を「分裂する “すごろくのコマ” 」で１つずつ進める．
その際，言語スコア，遷移スコア，尤度スコアを（対数上で）加算していく．



○ 赤色: “アクティブ”なノード

③ 探索空間のコンパクト化

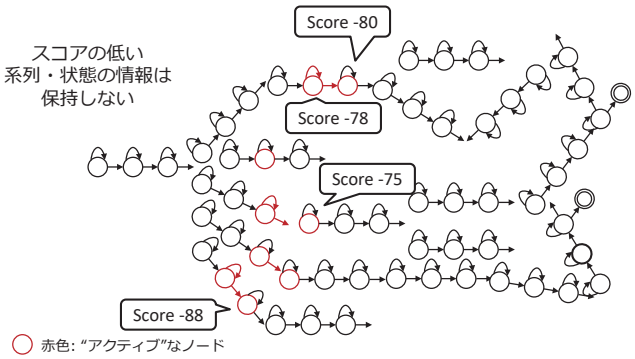
特徴量Nフレーム目の入力(枝刈りなし)



○ 赤色: “アクティブ”なノード

③ 探索空間のコンパクト化

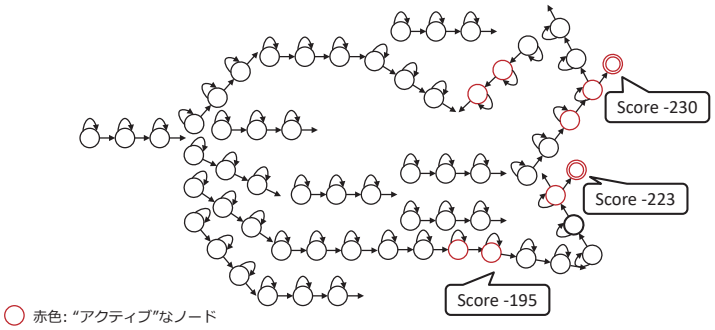
特徴量Nフレーム目の入力（枝刈りあり）



③ 探索空間のコンパクト化

特徴量最終フレーム目の入力

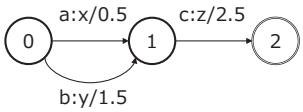
（理想的には）「文末」の状態（ノード）に存在している仮説＝受理すべきもの．
パスを逆にたどる(バックトラック) or メモったパスを参照し，「単語列」を出力．



WFST: Weighted Finite State Transducer

有限状態マシン

– 入力/出力シンボル，重み，状態で定義
– 入力シンボル列の受理と出力系列を評価可能



強力な演算 → コンパクトな探索空間を構築可

☺ WFSTの冗長性削除（状態数などの最小化）
☺ 2つのWFSTの合成

状態ネットワークの構築

大まかな作業の流れ

1. 各WFSTの構築
   - 単語 or N-gram WFST: 単語 → 単語の変換
   - 発音 WFST: 音素 → 単語の変換
   - HMM WFST: HMM状態 → 発音の変換
2. 各WFSTを合成・最適化（tool: OpenFST）

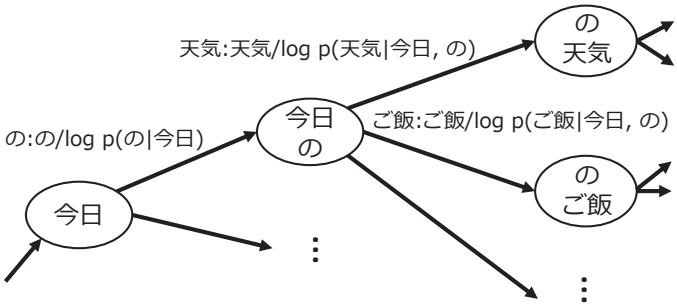
○ 合成演算

WFST: 単語 N-gram

$$p(\text{天気} \mid \text{今日}, \text{の})$$

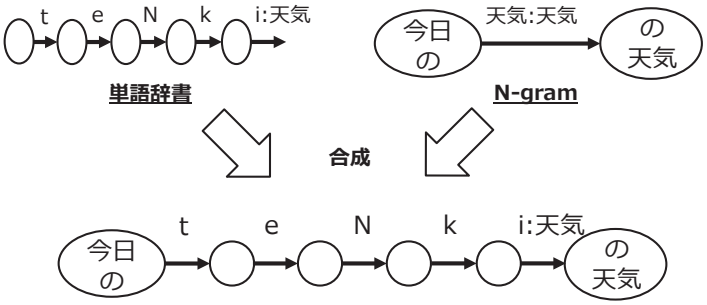
N-gram 確率の条件部分が状態
N-gram 確率（対数）が “重み”

WFST: 単語 N-gram



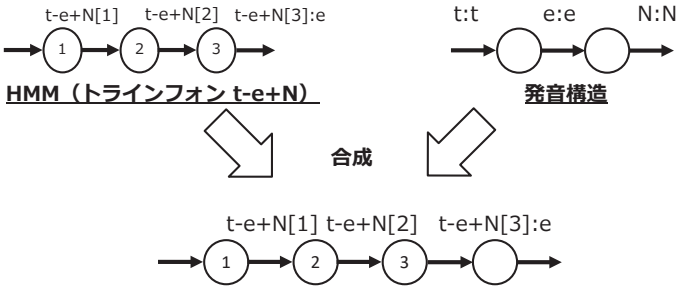
1-gram, 2-gram, … と連結

WFST: 発音 → 単語



発音辞書の合成

WFST: HMM → 発音



HMM構造の合成

WFST の規模感

	N-garm	Pruning 有無	アーク数	ノード数
単語	3	--	19,224,912	8,886,764
	3	✓	11,900,930	5,684,430
音素	8	✓	9,028,531	1,525,901
	5	✓	3,269,557	568,224
	3	✓	342,751	59,737
音節接続	--	--	75,958	19,225

単語認識: ノード数500万～のグラフ上の探索
処理量と精密度はトレードオフ
– Pruining: N-gram 確率が低い遷移を除去

END-TO-END 音声認識

End-to-End音声認識

End-to-End モデル

入力から出力への写像をそのまま学習

入力: 音響特徴量の系列 $\mathbf{x}_{1:T}=[\mathbf{x}_1,...,\mathbf{x}_T]$
出力: 音素/文字/単語等の記号列 $c_{1:L}=[c_1,...,c_L]$
(1 hot vector; embedding)

- 最終層は大体 softmax 関数

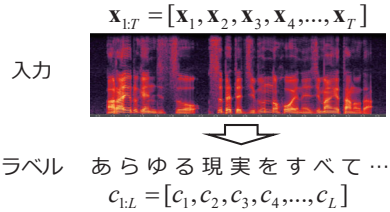
$$\hat{c}_{1:T}=\arg\max_{c_{1:T}} p(c_{1:T}\mid \mathbf{x}_{1:T})$$

ここを直接モデル化

End-to-End音声認識

学習時の設定

入力: 音響特徴量の系列 $\mathbf{x}_{1:T}=[\mathbf{x}_1,...,\mathbf{x}_T]$
ラベル: 音素/文字/単語等の記号列 $c_{1:L}=[c_1,...,c_L]$
(長さは fix)



添え字に注意
時間スケールが異なる

このズレの吸収が必要

End-to-End音声認識: I

CTC (Connectionist Temporal Classification)

$$p(c_{1:T} \mid \mathbf{x}_{1:T}) = \sum_{\boldsymbol{\pi} \in \Omega(c_{1:T})} \prod_t p(\pi_t \mid \mathbf{x}_t)$$

$\Omega(c_{1:T})$ 記号列$c_{1:T}$のパス集合
$\boldsymbol{\pi}$ 空白込みのパス
π_t パス上の t での文字

– HMM なしで系列を変換
  • $\mathbf{x}_t$ に対して,π_t の事後確率を計算し、同じ記号に縮約されるものの総和を計算
  • 例: 以下の記号列は全て “hai” に集約

_ h _ _ _ _ _ a _ _ _ _ i _
_ h h _ _ _ _ a a _ _ ii _
_ h _ _ _ a a a a a _ i i i

π_1 π_t π_{t+1}

“hai”に縮約される系列の尤度計算
⇒ HMM の尤度計算と同様

”_” は空白記号

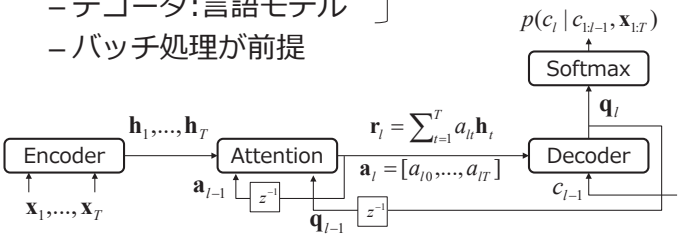
End-to-End音声認識: II

注意機構 (Attention) モデル

$$p(c_{1:l} \mid \mathbf{x}_{1:T}) = \prod_l p(c_l \mid c_{1:l-1}, \mathbf{x}_{1:T}) \quad \text{（自己回帰的）}$$

– エンコーダ:音響モデル
– デコーダ:言語モデル
– バッチ処理が前提

に概ね対応



そのほかのトピック

ダイアリゼーション
– （長時間の）録音データから，「誰がいつ話したのか」などを推定するタスク
– 音声区間検出，話者クラスタリング, おーばラップ区間処理, etc… を行う必要
→ End-to-end アプローチも適用

自己教師あり学習モデル: 音響的なモデル
– 大量のラベルなしデータから汎用的なネットワーク（特徴表現）を学習
– Denoising, 自己回帰的な予測などのタスク

全体の補足

ニューラル時代でも生き残っているもの
– **<u>問題(タスク)設定・本質的な問題</u>**
  • （潜在的に）重要な問題・課題,（意義のある)新しい問題/立て付けを発見することの価値
– **<u>基本的な, モデル・概念・理論・確率モデル</u>**
  • 物事や事象の性質・本質・メカニズムを捉えたもの
– **<u>学習・探索などにおけるアルゴリズム</u>**
  e.g. SGD, ビタビアルゴリズム, ビームサーチ, etc..
– （一部の）特徴量の知見
  e.g. メルフィルタバンク特徴量
  – 膨大なデータ利用時にどうなるかは…?

参考文献

- C.M.ビショップ「パターン認識と機械学習 上下」
- 河原達也 「音声認識システム」
- 安藤彰男 「リアルタイム音声認識」
- 岡崎, 他「自然言語処理の基礎」
- 増村 亮: “深層学習に基づく言語モデルと音声言語理解”, 日本音響学会誌 73 巻 1 号（2017），pp. 39–46. https://www.jstage.jst.go.jp/article/jasj/73/1/73_39/_pdf