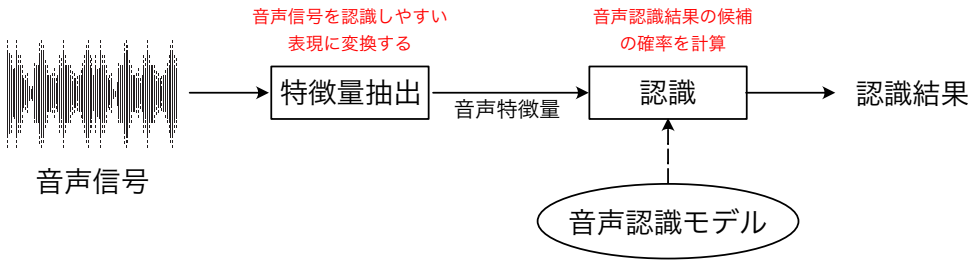
# CTC (Connectionist Temporal Classification)

音声認識 (ASR: Automatic Speech Recognition) とは、入力信号を音声特徴ベクトルに変換、その音声特徴ベクトルの系列から対応する単語列を推定するタスクです。近年、深層学習技術の進展に伴い音声認識モデルの精度は向上しており、様々な製品に応用されるようになっています。本解説プリントでは、そのような音声認識モデルの 1 つである **Connectionist Temporal Classification (CTC)** [Graves+,2006] について解説していきます。

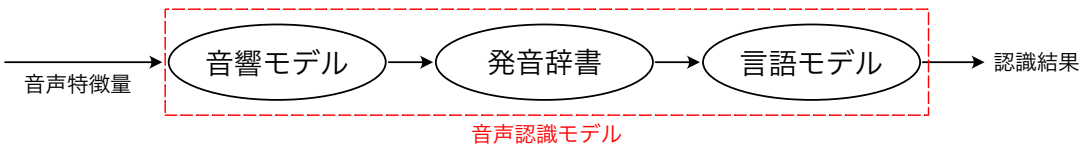
## 1　音声認識の概要

はじめに、機械がどのように音声認識を行っているか説明します。基本的な音声認識処理の流れは、下図のようになっています。




まず、入力された音声信号を認識処理にとって都合の良い表現に変換します。この処理のことを**特徴量抽出**と呼びます。この特徴量抽出では、主にフーリエ変換と呼ばれる数学的なテクニックを用いて音声信号を周波数ごとの音の強度に分解します。他にも人間の聴覚器官・発声器官の生物学的知見に基づく様々な処理が施されます[1]。

次に、認識の部分で「この音声は “天気” と言っている確率が 0.8 である」というように、音声認識結果の候補の確率を計算します。これは我々人間もよく行っている作業です。例えば、相手が言っていることがはっきりと聞こえなかったとしても、「今、○○と言ったのかな？それとも××かな？会話の流れ的に○○の可能性が高そうだな」のように無意識的に候補の可能性や妥当性を比べることで相手の言ったことを判断できる場合があります[2]。機械も同様に、確率という軸でそれぞれの候補の妥当性を評価し、最も確率の高い候補を認識結果として出力します。そして、この各候補の確率を計算する規則のことを**音声認識モデル**と呼びます。現在、商用化されている多くの音声認識モデルは、下図のような構造になっています。




[1] この部分の解説は本プリントでは省略します。
[2] 機械が誤認識してしまうように、人間もこれで勘違いしてしまうこともありますが…

音響モデル、発音辞書、言語モデルの 3 つの部分から構成されており、それぞれ以下の役割を担っています。

- **音響モデル**
  音声特徴量と音素列の間の確率を計算するモデル。音素とは、/a/や/i/といった母音、/k/や/s/といった子音から構成される音の最小単位のことです。音響モデルは、音を聴いて「これは “あ (/a/)”かな？それとも “い (/i/)”かな？」といった音素ごとの確率を求めます。

- **発音辞書**[3]
  音素列と単語との対応を扱うモデル。辞書という名前の通り、「おはよう：/o/h/a/y/o」のような単語とその発音 (音素列) が記述されたリストとなっています。

- **言語モデル**
  ある単語に対して、その単語が発話される確率を計算します。例えば、1 つ前の単語が「いい」だとすると、次の単語は「電気」よりも「天気」の方が来やすく (=確率が高く) なります。つまり、文脈的にその単語が表れやすいかの確率を計算するモデルとなっています。

以上が (従来的な) 音声認識処理のざっくりとした構造です。音響モデルについては、1980 年代から 2010 年ごろまでの長い間、**隠れマルコフモデル (HMM: hidden Markov model)** と**混合正規分布 (GMM: Gaussian mixture model)** と呼ばれるモデルを組み合わせることでモデル化されていました。それが 2010 年ごろから深層学習技術の発展に伴い、混合正規分布がディープニューラルネットワーク (DNN: Deep Neural Network) に置き換わることで音声認識精度が飛躍的に向上し、多くの製品で使われる音声認識モデルとなっています。

このように音声認識モデルを 3 つのモジュール (音響モデル、発音辞書、言語モデル) に分割する手法は大きな成功を収めているのですが、1 つ大きな問題があります。それは直感的には非常に理解しやすい構造なのですが、実際に実装する際には非常に複雑になってしまう点です。特に、3 つのモジュールの出力をうまく統合して音声認識結果を出力する「デコーダ」と呼ばれる処理部分の実装が難しく、音声認識モデルを実装する上で高いハードルとなっています。

このような問題に対し、2015 年ごろからは音響モデル、発音辞書、言語モデルを 1 つの DNN で表現する、いわゆる End-to-End モデル (E2E モデル) の研究が盛んになっています。End-to-End モデルでは複数のモジュールの統合といった処理は考える必要がないため、構成がシンプルで比較的簡単に実装できるという魅力があります。この解説プリントのメインテーマである CTC は、このような End-to-End モデルの 1 つとなっています[4]。

[3] パターンマッチ辞書モデルなどと呼ばれることもあります。

[4] CTC の他にも、Attention モデルが E2E モデルとしてよく知られています。なお、CTC は言語モデルに相当する部分が明示的にモデル化されていないため、一般には言語モデルとの併用が推奨されています。そのため、CTC は E2E モデルに含めないという意見もあります。

# 2　CTC (Connectionist Temporal Classification)

音声認識の基本的な知識・用語を整理したとこで、本題の CTC に入りたいと思います。CTC は End-to-End モデルの中でも比較的初期に提案されたモデルで、従来手法のように隠れマルコフモデル (HMM) を使用せずにディープニューラルネットワーク (DNN) だけで音響モデルを構築する手法として提案されました [Graves+,2006]。従来の 3 つのモジュールに分割する手法では DNN の出力を HMM を用いてラベル (音素) 列の確率に変換する必要がありましたが、CTC では HMM を介さずに直接ラベル列の確率を出力することが可能になっています。

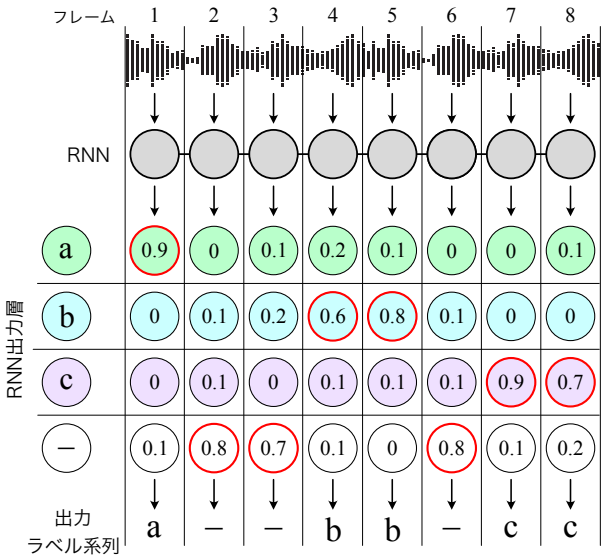
CTC における重要な発明は次の 2 点です。

- ブランク (blank) と呼ばれるラベルの導入[5]
- 前向き・後ろ向きアルゴリズム (forward-backward algorithm) を用いた DNN の学習

以下ではこの 2 つを中心に CTC の基礎事項を解説していきます。なお、CTC では基本的には音声のような時系列情報を扱うため、**RNN (recurrent newral network)** や **LSTM (Long Short Term Memory)** のような時系列を考慮した DNN を用います[6]。そのため、以下では RNN を前提として CTC を考えていくことにします。

## 2.1　ブランクの導入

RNN に音声系列を入力すると、フレーム数だけ出力が得られます。例として 8 フレームの音声系列を RNN に入力し、出力値 (確率)[7]が最も高いラベル (音素) をフレーム毎に出力した結果、[a, −, −, b, b, −, c, c] というラベル系列が得られた場合を考えてみましょう。

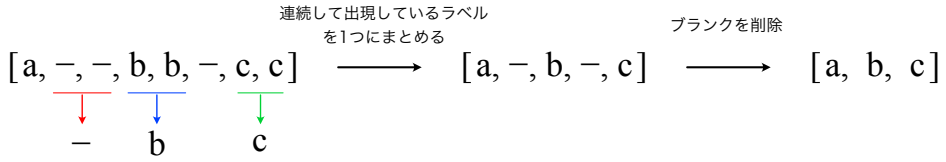



[5]「冗長ラベル」などとも呼ばれます。
[6]RNN の詳しい解説はこのプリントでは省略します。
[7]以下では softmax 関数が適用された出力と考え、出力値を確率と解釈します。

ここで「–」はブランクを表しています。CTC では、次の手順でこのフレーム単位のラベル系列を最終的なテキスト系列に変換します (この変換のことを**縮約**と呼びます)。

1. 連続して出現している同一ラベルを 1 つのラベルにまとめる[8]
2. ブランク「–」を削除する

[a, –, –, b, b, –, c, c] を上記の手順に従って縮約すると、次のようになります。




以下では提案論文の記法に従い、この縮約を関数 $\mathcal{B}$ で表すことにします。

$$\mathcal{B}(\mathrm{a}, -, -, \mathrm{b}, \mathrm{b}, -, \mathrm{c}, \mathrm{c}) = \mathrm{a}, \mathrm{b}, \mathrm{c}$$

なぜ最終的には削除するブランクを、CTC ではわざわざ導入するのでしょうか？ブランクを導入する理由は大きく 2 つあります。1 つは、同一ラベルが連続するテキスト系列を表現するためです。ブランクが存在しない場合、例えば [a, a, a, a, b, b, b] は [a, b] に縮約されてしまうため、[a, a, b] を表現することができません。連続するラベルの間に [a, a, –, a, a, b, b] のようにブランクが挿入されることにより、同一ラベルが連続するようなテキスト系列も表現できるようになります。もう 1 つの理由は、厳密にアライメントを決定させないためです。音素の境界は曖昧なことも多く、また単語の間にはポーズ (間) などの非音声区間も存在します。そのようなフレームに対しても何らかのラベルを無理やり割り当てようとすると、音声認識にとって最適ではないモデルが学習される可能性があります。そこで、該当するラベルがないことを意味するブランクの出力を許可することで、モデルが無理なアライメント推定を行わずに音声認識結果が正解することのみを目的とした学習を可能にしています。

## 2.2 前向き・後向きアルゴリズムを用いた RNN の学習 (1)

さて、先ほどの 8 フレームの音声系列の例で、最終的なテキスト列が [a, b, c] となるような RNN の出力は [a, –, b, b, b, c, –, –]、[–, –, a, –, b, b, –, c] など沢山存在します。

$$\mathcal{B}(\mathrm{a}, -, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{c}, -, -) = \mathrm{a}, \mathrm{b}, \mathrm{c}$$
$$\mathcal{B}(-, -, \mathrm{a}, -, \mathrm{b}, \mathrm{b}, -, \mathrm{c}) = \mathrm{a}, \mathrm{b}, \mathrm{c}$$
$$\vdots$$

[8] これにより、例えば音声データの中で文字が伸びて発音されていた場合でも、文字列を繰り返すことで表現して最終的には 1 つの文字にまとめて出力することが可能となります。

つまり、入力音声系列 $\boldsymbol{x}$[9]に対して縮約後の出力テキスト系列が $\boldsymbol{l} = [a, b, c]$ となる事後確率は、

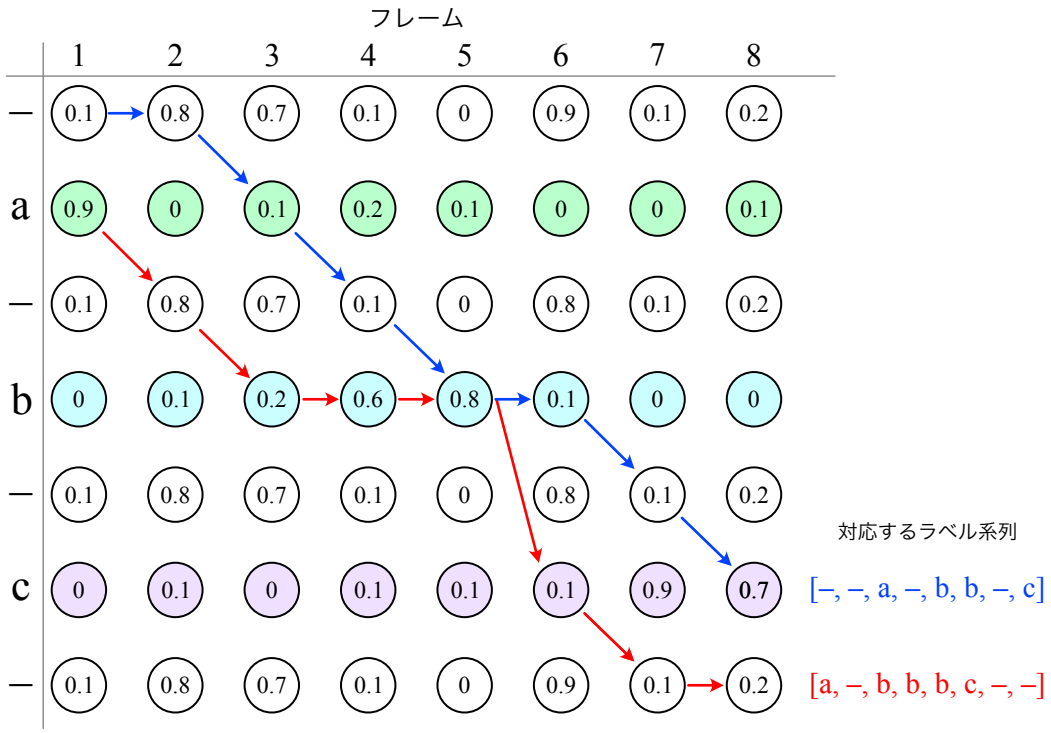
$$
\begin{aligned}
P(\boldsymbol{l}|\boldsymbol{x}) &= P\big([\mathrm{a}, -, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{c}, -, -]|\boldsymbol{x}\big) + P\big([-, -, \mathrm{a}, -, \mathrm{b}, \mathrm{b}, -, \mathrm{c}]|\boldsymbol{x}\big) + \cdots \\
&= \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{l})} P(\boldsymbol{\pi}|\boldsymbol{x}) \qquad (1)
\end{aligned}
$$

のような総和で表現されます。ここで、$P\big([\mathrm{a}, -, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{c}, -, -]|\boldsymbol{x}\big)$ は入力 $\boldsymbol{x}$ に対して RNN の (縮約前の) 出力が $[\mathrm{a}, -, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{c}, -, -]$ となる確率を表します。また $\mathcal{B}^{-1}(\boldsymbol{l})$ は「縮約するとテキスト系列 $\boldsymbol{l}$ になるような縮約前のラベル系列の集合」を表しています。この集合に含まれるラベル系列 $\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{l})$ は、[a, −, b, b, b, c, −, −]、[−, −, a, −, b, b, −, c] など複数存在するため、それら全てを足し合わせる ($\sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{l})}$) ということを上式は意味しています。

音声認識モデルは、この $P(\boldsymbol{l}|\boldsymbol{x})$ が最大となるようなテキスト系列 $\boldsymbol{l}$ を音声認識結果として出力します。ですから学習の際には、正解テキスト系列 $\boldsymbol{l}^*$ における確率 $P(\boldsymbol{l}^*|\boldsymbol{x})$ が最大となるように RNN のパラメータ調整を行うことになります。つまり、CTC において最小化すべき損失関数は、この事後確率 $P(\boldsymbol{l}^*|\boldsymbol{x})$ の対数に $-1$ をかけたものになります。

$$
\mathcal{L}_{\mathrm{CTC}} = -\log P(\boldsymbol{l}^*|\boldsymbol{x}) \qquad (2)
$$

では、具体的にどのように $P(\boldsymbol{l}^*|\boldsymbol{x})$ は計算されるのでしょうか？以下では、入力音声系列 $\boldsymbol{x}$ に対する正解テキスト系列が $\boldsymbol{l}^* = [a, b, c]$ である場合を考えてみましょう。$P(\boldsymbol{l}^*|\boldsymbol{x})$ を計算する際には、次のような絵 (グラフ) を考えると分かりやすいでしょう。




[9] より具体的には、特徴量抽出を行った後の音声特徴量ベクトルとなります。

このグラフでは、フレーム 1 の始点から次のフレームの頂点 (丸) へと順々に遷移しており、辿り着いた頂点に対応するラベルがラベル系列として出力されると考えます。例えば、先ほどの [a, −, b, b, b, c, −, −] というラベル系列は赤色の矢印に、[−, −, a, −, b, b, −, c] のラベル系列は青色の矢印に対応します。このようにラベル系列を矢印で表したもののことを**パス (path)** と呼びます。なお、白い頂点はブランクを表していますが、同一フレームにおける (4 つの) ブランクは全て同じものを表しています。また、上図で各頂点の数字は RNN の出力値 (確率) を表しています[10]。以下では、提案論文の記法に従って、この確率を $y_k^t$ と表すことにします。$y_k^t$ とは、フレーム $t$ $(= 1, 2, \cdots, 8)$ でラベル $k$ $(= \mathrm{a}, \mathrm{b}, \mathrm{c}, -)$ が出力される確率を表しています。上図の例では $y_{\mathrm{a}}^1 = 0.9$、$y_{-}^5 = 0$ といった具合になります。

$P(\boldsymbol{l}^*|\boldsymbol{x})$ を計算することは、上図で赤や青のパスような縮約すると $\boldsymbol{l}^* = [a, b, c]$ となる全てのパスの確率を足し合わせることに対応します。それぞれのパスの確率自体は、パス上で各ラベルが出力される確率の積として簡単に計算できます。例えば赤色のパスの場合には、次のように計算できます。

$$\begin{aligned} P\big([\mathrm{a}, -, \mathrm{b}, \mathrm{b}, \mathrm{b}, \mathrm{c}, -, -]|\boldsymbol{x}\big) &= y_{\mathrm{a}}^1 \times y_{-}^2 \times y_{\mathrm{b}}^3 \times y_{\mathrm{b}}^4 \times y_{\mathrm{b}}^5 \times y_{\mathrm{c}}^6 \times y_{-}^7 \times y_{-}^8 \\ &= 0.9 \times 0.8 \times 0.2 \times 0.6 \times 0.8 \times 0.1 \times 0.1 \times 0.2 \end{aligned} \tag{3}$$

このような計算を他のパスに関しても行えば良いだけです。つまり、

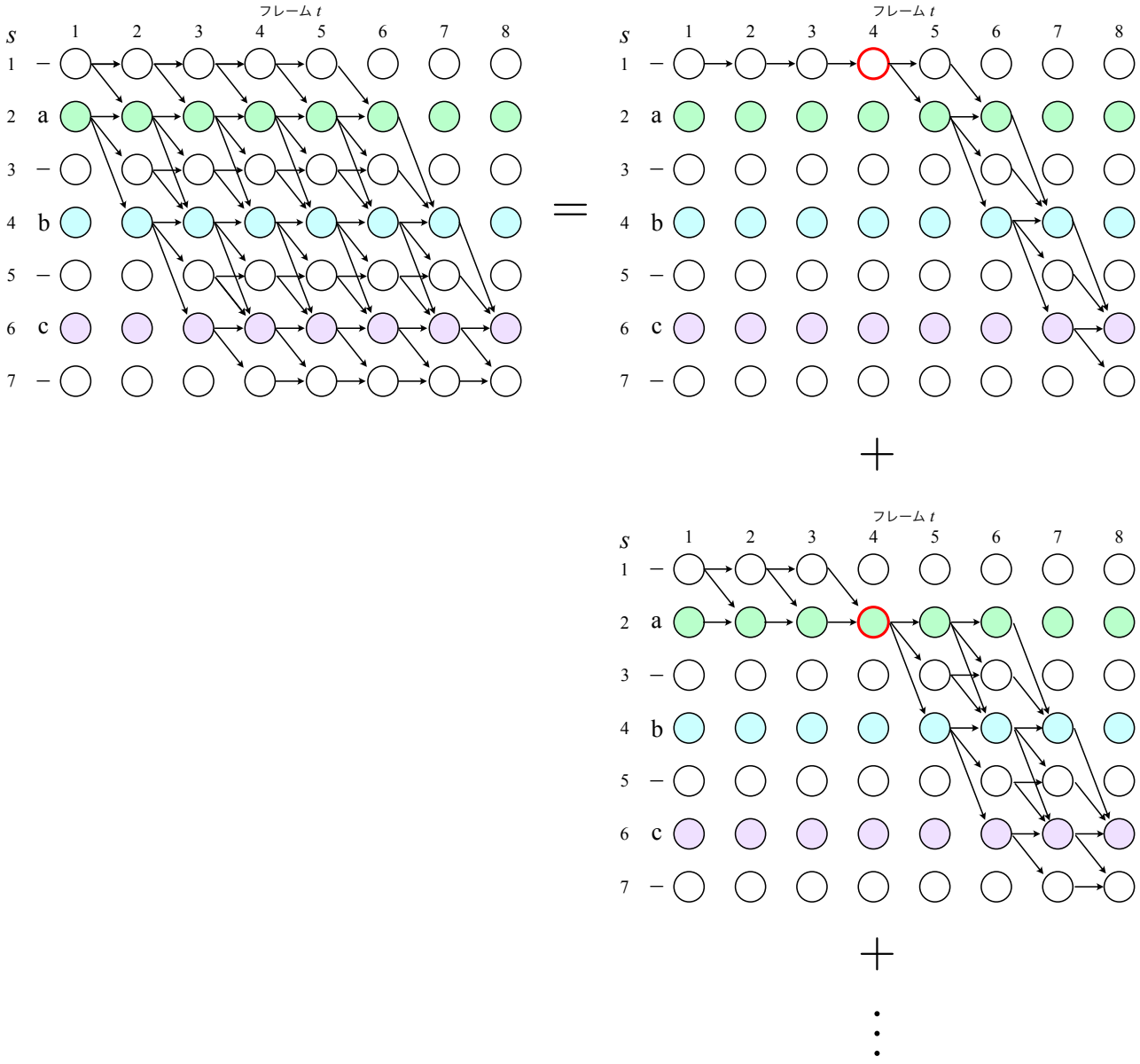
$$P(\boldsymbol{l}^*|\boldsymbol{x}) = \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{l}^*)} P(\boldsymbol{\pi}|\boldsymbol{x}) = \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{l}^*)} \prod_{t=1}^{T} y_{\pi_t}^t \tag{4}$$

と計算できます[11]。しかし、少し考えれば分かりますが縮約して $\boldsymbol{l}^* = [a, b, c]$ となるパスは大量にあるため、全てを愚直に計算するのは非効率です。そのため実際の CTC では、より効率的な計算方法である前向き・後ろ向きアルゴリズム (forward-backward algorithm) が用いられます。以下ではこのアルゴリズムの概要を説明していきます。

まず、$\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{l}^*)$ なるパスを全て同時に考えるとややこしいので、いくつかのグループに分類します。具体的には、下図のようにあるフレーム $t$ (下図では $t = 4$) においてどの頂点を通るかに注目して分類します。ここでは頂点の位置を表す $s$ という変数を導入しました。そして、$\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{l}^*)$ なるパスを $(t, s) = (4, 1)$ を通るもの、$(t, s) = (4, 2)$ を通るもの、$\cdots$、といった具合に分類しています。フレーム 1 の始点からフレーム 8 の終点に辿り着くためには、必ずフレーム 4 のどこかの頂点を通る必要があります。そしてある 1 つのパスが通るフレーム 4 の頂点はどれか 1 つのみです [例えば $(t, s) = (4, 1)$ と $(t, s) = (4, 2)$ の両方を通るようなパスは存在しません]。そのため、このようにあるフレームでどの頂点を通るかに注目することで、漏れや重複なくパスを分類することができます。

[10] 同一フレームの 4 つのブランクは同じものを表すため、確率の値も同じになっています。

[11] $T$ はフレーム数、今の例の場合は $T = 8$ です。また、$\pi_t$ はパス $\boldsymbol{\pi}$ のフレーム $t$ におけるラベルを表します。




確率の言葉で言えば、この分類はいわゆる「排反な事象への分類」になっており、次の確率の和の法則が成り立ちます。

$$
\begin{aligned}
P(\boldsymbol{l}^*|\boldsymbol{x}) &= \sum_{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{l}^*)} P(\boldsymbol{\pi}|\boldsymbol{x}) \\
&= \sum_{\substack{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{l}^*),\\ \pi_{t=4}=l'_1}} P(\boldsymbol{\pi}|\boldsymbol{x}) + \sum_{\substack{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{l}^*),\\ \pi_{t=4}=l'_2}} P(\boldsymbol{\pi}|\boldsymbol{x}) + \sum_{\substack{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{l}^*),\\ \pi_{t=4}=l'_3}} P(\boldsymbol{\pi}|\boldsymbol{x}) + \cdots \\
&= \sum_{s=1}^{|\boldsymbol{l}'|} \sum_{\substack{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{l}^*),\\ \pi_{t=4}=l'_s}} P(\boldsymbol{\pi}|\boldsymbol{x}) \qquad (5)
\end{aligned}
$$

ここでいくつか新しい文字を導入しました。まず拡張ラベル $\boldsymbol{l}' = (l'_1, l'_2, \cdots)$ は、正解ラベル

$\boldsymbol{l}^*$ の各ラベルの前後にブランクを挿入したラベル系列です。つまり、今の例の場合には $\boldsymbol{l}' = [-, \mathrm{a}, -, \mathrm{b}, -, \mathrm{c}, -]$ であり、$l'_1 =$“$-$”、$l'_2 =$“a”、$l'_3 =$“$-$”、$\cdots$、といった具合になっています。$\sum_{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{l}^*),\ \pi_{t=4}=l'_1}$ というのは、$\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{l}^*)$ なるパスのうち、$(t, s) = (4, 1)$ を通るパスのみについて和を取るという意味です。ですから式 (5) の 2 行目は、先ほど言葉で説明したパスの分類を数式で表現しただけです。そして、3 行目は 2 行目の内容を総和記号を用いてまとめています。$|\boldsymbol{l}'|$ は拡張ラベルの要素数を表していて、今の例の場合には $|\boldsymbol{l}'| = 7$ となっています。

ここで注意してもらいたいのは、今は $t = 4$ のフレームに注目しましたが、上記の議論は任意の $t$ に対して成り立つということです。別にどのフレームに注目しても同様な「排反な事象への分類」が可能であるため、式 (5) は任意の $t$ に対して成り立つ等式となっています。つまり、

$$P(\boldsymbol{l}^*|\boldsymbol{x}) = \sum_{s=1}^{|\boldsymbol{l}'|} \sum_{\substack{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{l}^*),\\ \pi_t=l'_s}} P(\boldsymbol{\pi}|\boldsymbol{x}) \quad (\text{for any } t) \tag{6}$$

となります。

さて、パスを分類したことで見通しがよくなりましたが、そもそも

$$\sum_{\substack{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{l}^*),\\ \pi_t=l'_s}} P(\boldsymbol{\pi}|\boldsymbol{x}) \qquad (s = 1, 2, \cdots, |\boldsymbol{l}|')$$

はどのように計算すれば良いのでしょうか？ CTC では、上の量の計算のために次の 2 つの確率を導入します[12]。

- 前向き確率 $\alpha_t(s)$

  始点からフレーム $t$、拡張ラベル $s$ の頂点に到達するまでの全パスの確率の総和

$$\alpha_t(s) \equiv \sum_{\mathcal{B}(\boldsymbol{\pi}_{1:t})=\boldsymbol{l}^*_{1:[s/2]}} \prod_{t'=1}^{t} y^{t'}_{\pi_{t'}} \tag{7}$$

- 後ろ向き確率 $\beta_t(s)$

  フレーム $t$、拡張ラベル $s$ の頂点から終点まで到達する全パスの確率の総和

$$\beta_t(s) \equiv \sum_{\mathcal{B}(\boldsymbol{\pi}_{t:T})=\boldsymbol{l}^*_{[s/2]:|\boldsymbol{l}^*|}} \prod_{t'=t}^{T} y^{t'}_{\pi_{t'}} \tag{8}$$

上式で導入した記号について整理しておきます。$\boldsymbol{a}_{i:j}$ は、成分 $i$ から成分 $j$ までを持つベクトル (系列) を表しています。$[\cdot]$ はガウス記号を表しています。例えば、$s = 3$ の時には $[s/2] = [1.5] = 1$

[12] 記法は提案論文 [Graves+,2006] に従っています。ただ、提案論文では縮約前のラベル系列 $\boldsymbol{l}'$ と縮約後の正解ラベル系列 $\boldsymbol{l}^*$ の要素を表す変数 $s$ が混同されて用いられていると思われるため、(細かい点なので気にしなくて大丈夫ですが) その部分はガウス記号を用いた記法に変更しています。

となります。そのため、例えば$t=4$、$s=4$に対しては

$$\mathcal{B}(\boldsymbol{\pi}_{1:4}) = \boldsymbol{l}^*_{1:[4/2]} = \boldsymbol{l}^*_{1:2} = [\mathrm{a}, \mathrm{b}]$$

となります。つまり、$\mathcal{B}(\boldsymbol{\pi}_{1:t=4}) = \boldsymbol{l}^*_{1:[s=4/2]}$は縮約するとラベル系列 [a, b] となるフレーム 1∼4 の頂点を通るパス$\boldsymbol{\pi}_{1:4}$の集合を表しています。この前向き確率と後ろ向き確率を掛け合わせると、

$$\begin{aligned}
\alpha_t(s)\beta_t(s) &= \left( \sum_{\mathcal{B}(\boldsymbol{\pi}_{1:t})=\boldsymbol{l}^*_{1:[s/2]}} \prod_{t'=1}^{t} y^{t'}_{\pi_{t'}} \right) \left( \sum_{\mathcal{B}(\boldsymbol{\pi}_{t:T})=\boldsymbol{l}^*_{[s/2]:|\boldsymbol{l}|}} \prod_{t'=t}^{T} y^{t'}_{\pi_{t'}} \right) \\
&= \sum_{\mathcal{B}(\boldsymbol{\rho}_{1:t})=\boldsymbol{l}^*_{1:[s/2]}} \sum_{\mathcal{B}(\boldsymbol{\sigma}_{t:T})=\boldsymbol{l}^*_{[s/2]:|\boldsymbol{l}|}} \prod_{\tau=1}^{t} y^{\tau}_{\rho_\tau} \prod_{\nu=t}^{T} y^{\nu}_{\sigma_\nu} \\
&= y^t_{l'_s} \sum_{\substack{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{l}^*),\\ \pi_t=l'_s}} \prod_{t'=1}^{T} y^{t'}_{\pi_{t'}} \\
&= y^t_{l'_s} \sum_{\substack{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{l}^*),\\ \pi_t=l'_s}} P(\boldsymbol{\pi}|\boldsymbol{x}) \qquad (9)
\end{aligned}$$

となります。最後の等式に移るところで式 (4) を用いています。これより、フレーム$t$で拡張ラベル$s$の頂点を通るパスの確率の総和が

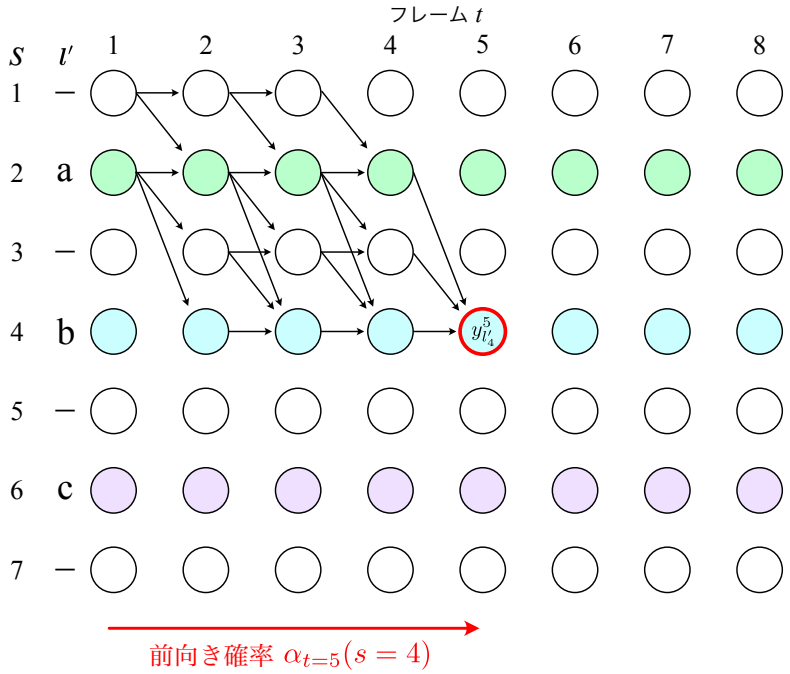
$$\sum_{\substack{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\boldsymbol{l}^*),\\ \pi_t=l^*_s}} P(\boldsymbol{\pi}|\boldsymbol{x}) = \frac{\alpha_t(s)\beta_t(s)}{y^t_{l^*_s}} \qquad (10)$$

と表せることが分かります。結局、上式を式 (6) に代入することで次式が得られます。

$$P(\boldsymbol{l}^*|\boldsymbol{x}) = \sum_{s=1}^{|\boldsymbol{l}^*|} \frac{\alpha_t(s)\beta_t(s)}{y^t_{l^*_s}} \quad (\text{for any } t) \qquad (11)$$

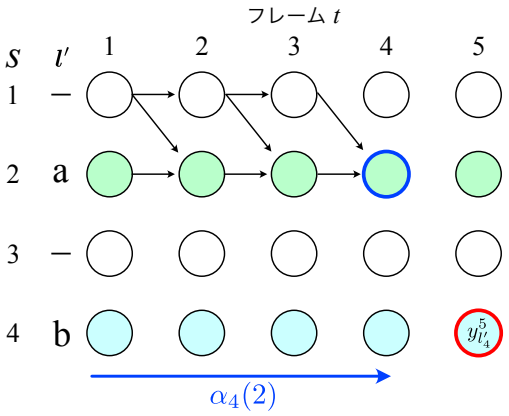
つまり、$P(\boldsymbol{l}^*|\boldsymbol{x})$ [そして CTC 損失関数 $\mathcal{L}_{\mathrm{CTC}} = -\log P(\boldsymbol{l}^*|\boldsymbol{x})$] を計算するためには、前向き確率$\alpha_t(s)$と後ろ向き確率$\beta_t(s)$さえ用意すればよいことが分かります。

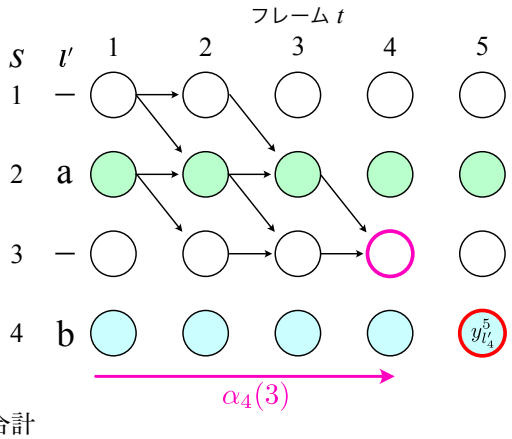
前向き、後ろ向き確率を導入してさらに見通しは良くなりましたが、それらの確率はどのように計算されるのでしょうか?以下では$\alpha_t(s)$と$\beta_t(s)$の計算の仕方を説明します。実は、これらの確率は再帰的に計算することができる構造になっており、これにより効率的に$P(\boldsymbol{l}^*|\boldsymbol{x})$を計算することが可能になっています。このことを見るために、先ほどまでの例で具体的に$\alpha_{t=5}(s=4)$を考えてみましょう。$\alpha_5(4)$は、フレーム$t=5$で拡張ラベル$s=4$ (つまり “b”) の頂点に辿り着く全パスの確率の和でした。考える全パスは下図のようになります。

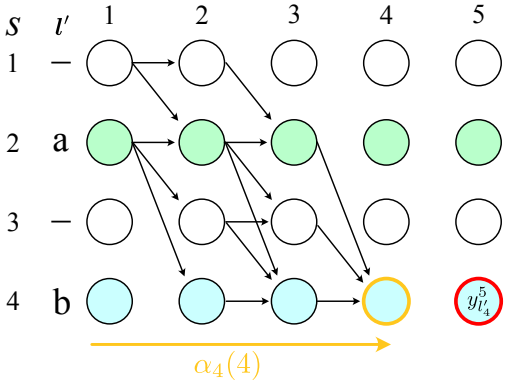



$\alpha_5(4)$ は、上記の全てのパスの確率の和を足せば確かに求まるのですが、次のようフレーム $t=4$ での前向き確率を用いて表現することもできます：

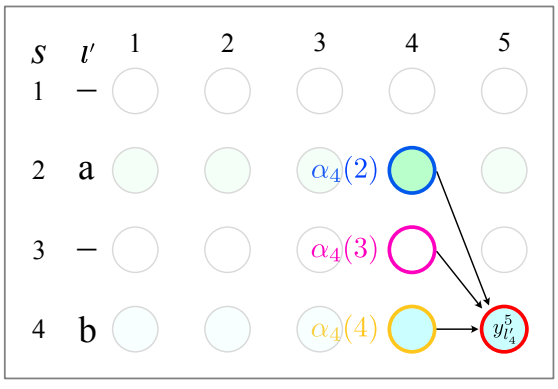
$$\alpha_5(4) = \left[\alpha_4(2) + \alpha_4(3) + \alpha_4(4)\right] \times y^5_{l'_4} \tag{12}$$




合計



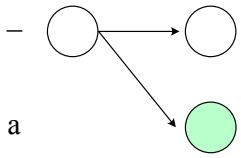
これは、上図のように分解したことに相当します。つまり、フレーム $t$ の前向き確率はフレーム $t-1$ の前向き確率のみから簡単に計算できるのです。フレーム 5 での前向き確率が知りたいならば、フレーム 1 からいちいちパスを考える必要はなく、フレーム 4 での前向き確率さえ分かっていればいいのです。言い換えれば、前向き確率に関する漸化式が存在して、フレーム 1 での前向き確率さえ与えれば、あとはその漸化式に従って次々と次のフレームの前向き確率 $\alpha_t(s)$ が効率的に計算できる構造になっています。

では、その前向き確率 $\alpha_t(s)$ に関する一般的な漸化式を考えてみましょう。まず初項 (フレーム 1) ですが、始点はブランク $(s=1)$ か正解ラベルの 1 文字目 $\boldsymbol{l}^*_1$ $(s=2)$ で 2 通り存在します。そのため、前向き確率の初項は次にように与えられます。

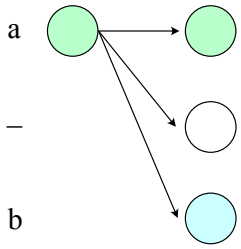
$$\alpha_0(s) = \begin{cases} y^0_- & (s=1) \\ y^0_{l^*_1} & (s=2) \\ 0 & (s>1) \end{cases} \tag{13}$$

次に、フレーム $t$ からフレーム $t+1$ $(t \geq 1)$ への遷移を考えます。落ち着いて考えると分かりますが、前向き確率で許される遷移としては次の 3 つのパターンがあることが分かります。少し

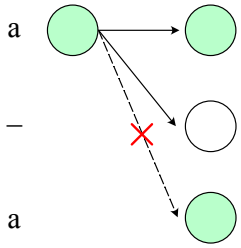
(1) ブランクからはブランクか次の文字へ遷移




(2) ブランクでない頂点からは、同じ文字か、ブランク、次の文字へ遷移




(3) ただし、同じ文字が続く場合は、必ずブランクを経由する




(ブランクを飛ばすと、縮約した時に[a, a]とならない)

注意が必要なのは (3) のようなケースです。正解ラベル系列で [a, a] のように同じ文字が連続する場合には、必ずブランクを経由する必要があります (そうでないと、縮約した時に [a] となっ

てしまうため)。以上を踏まえると、フレーム $t+1$ の前向き確率は、フレーム $t$ の前向き確率から次のように計算されることが分かります。

$$\alpha_{t+1}(s) = \begin{cases} \big[\alpha_t(s) + \alpha_t(s-1)\big] y_{l_s^*}^{t+1} & (\text{if } l_s^* = \text{blank or } l_{s-2}^* = l_s^*) \\ \big[\alpha_t(s) + \alpha_t(s-1) + \alpha_t(s-2)\big] y_{l_s^*}^{t+1} & (\text{otherwise}) \end{cases} \tag{14}$$

この $\alpha_t(s)$ の $t$ に関する漸化式を式 (13) の初項の下で解くことで、効率的に必要な $\alpha_t(s)$ が計算できます。尚、今は前向き確率に関して考えましたが、全く同じようなことが後ろ向き確率に関しても考えられます。同じことの繰り返しになりますので詳細は省略します、気になる方は提案論文 [Graves+,2006] の式 (10) などを参照して下さい。

## 2.3 前向き・後向きアルゴリズムを用いた RNN の学習 (2)

前のセクションでは前向き確率・後ろ向き確率の効率的な計算方法、そしてそれらを式 (11) に代入することで $P(\boldsymbol{l}^*|\boldsymbol{x})$ 、そして CTC 損失関数 $\mathcal{L}_{\text{CTC}} = -\log P(\boldsymbol{l}^*|\boldsymbol{x})$ が求まることを説明しました。実際の RNN の学習では、RNN の出力層に対する損失関数 $\mathcal{L}_{\text{CTC}}$ の勾配を計算し、その勾配を用いて誤差逆伝播 (バックプロパゲーション) を行うことでネットワークのパラメータを更新することになります。出力層のノード $y_k^t$ に対する CTC 損失関数の勾配は、次のように計算されます。

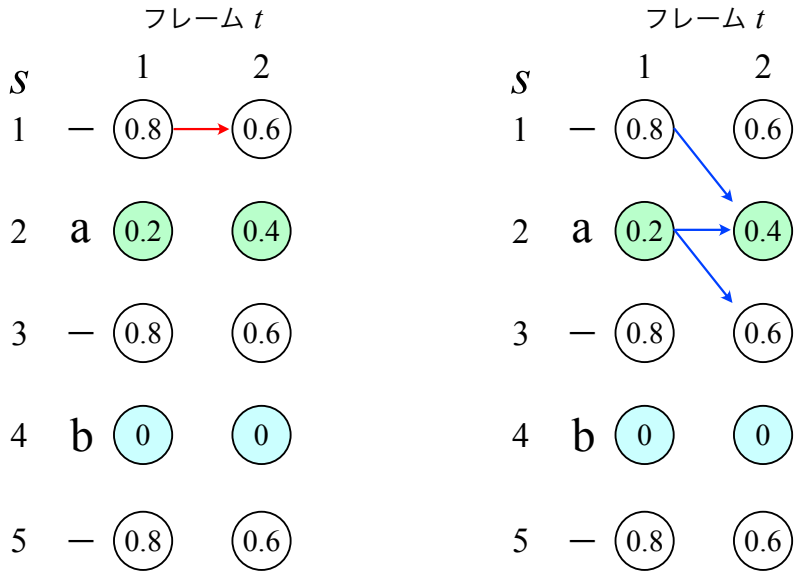
$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{CTC}}}{\partial y_k^t} &= -\frac{1}{P(\boldsymbol{l}^*|\boldsymbol{x})} \frac{\partial P(\boldsymbol{l}^*|\boldsymbol{x})}{\partial y_k^t} \\ &= -\frac{1}{P(\boldsymbol{l}^*|\boldsymbol{x})} \frac{\partial}{\partial y_k^t} \left( \sum_{s=1}^{|\boldsymbol{l}^*|} \frac{\alpha_t(s)\beta_t(s)}{y_{l_s^*}^t} \right) \\ &= -\frac{1}{P(\boldsymbol{l}^*|\boldsymbol{x})} \left( -\sum_{s \in lab(\boldsymbol{l}^*, k)} \frac{\alpha_t(s)\beta_t(s)}{\left(y_{l_s^*}^t\right)^2} \right) \\ &= \frac{1}{P(\boldsymbol{l}^*|\boldsymbol{x})} \frac{1}{(y_k^t)^2} \sum_{s \in lab(\boldsymbol{l}^*, k)} \alpha_t(s)\beta_t(s) \end{aligned} \tag{15}$$

2 つ目の等号で式 (11) を $P(\boldsymbol{l}^*|\boldsymbol{x})$ に代入しました。また、提案論文の表記に倣って $lab(\boldsymbol{l}^*, k)$ という記号を導入しました。$s \in lab(\boldsymbol{l}^*, k)$ とは、$\boldsymbol{l}_s^* = k$ となるような $s$ の集合を表しています。式 (15) より、$P(\boldsymbol{l}^*|\boldsymbol{x})$ だけでなく、CTC の損失関数の勾配も前向き・後ろ向き確率を用いて計算できることが分かります。つまり、基本的には漸化式を解いて $\alpha_t(s)$、$\beta_t(s)$ さえ求めれば、RNN を学習することができます。

## 2.4 CTC による音声認識

ここまでで RNN の学習方法を説明しました。最後に、学習されたネットワークを用いて CTC でどのように音声認識 (デコーディング) が行われるか簡単に説明します。CTC によるデコーディングの処理方法はいくつか存在しますが、ここでは **best path decoding** と呼ばれるシンプルな方法を取りあげます。

推論時には、学習時とはことなり正解ラベル系列 $\boldsymbol{l}^*$ は与えられていないため、仮説ごとに尤度を計算して比較を行うのが最も自然な発想となります。しかし、仮説数が多くなると計算量が多くなるため、愚直にこの比較を行うことは困難です。そこで best path decoding では、各フレームで最も出力値 (確率) が高い頂点を通るパス (best path) のみに注目し、この best path を縮約して得られるテキスト系列を認識結果として出力します。




例えば、上の例ではフレーム 1、2 共にブランクの確率が最も高くなっているため、赤色で示した $(s,t) = (1,1) \to (1,2)$ というパスが best path となります。ですので、best path decoding ではこの部分は「$\mathcal{B}(-,-) =$ ラベルなし」と認識されることになります。

best path decoding は非常にシンプルで高速なデコーディング方法ですが、問題もあります。それは、best path を縮約して得られるテキスト系列が必ずしも最も確率の高い出力とは限らない点です。例えば、上の例では best path の確率は $0.8 \times 0.6 = 0.48$ となっています。一方、縮約した結果が [a] となるようなパス (上図で青色で示した 3 つのパス) の確率の合計は、$0.2 \times 0.4 + 0.2 \times 0.6 + 0.8 \times 0.4 = 0.52$ となります。つまり、上の例では「ラベルなし」という認識結果の確率 (0.48) よりも「a」という認識結果の確率 (0.52) の方が大きいという状況になっています。このような状況にも対応するために、**beam search decoding** などのより複雑なデコーディング方法が実践的にはよく用いられます。こちらの内容は本解説プリントの範囲を超えるので割愛しますが、興味のある方は調べてみて下さい。