レポート課題 1：DP マッチングによる単語音声認識

授業中に解説した**動的計画法 (DP) マッチング**のアルゴリズムを利用し，小規模の単語音声認識実験を行う。簡単のため，音声入力→音響分析までの過程はすでに終了しているものとし，あらかじめ用意されたテキストファイルのデータを入力とする。100 単語の**テンプレート**に対して，同じ発声内容の 100 単語（同一話者または別話者）を未知音声と見立てて入力したとき，何単語が正しく認識できるかを調べる。

まず，データの取り扱いについて説明する。

(1) データファイル (`city_mcepdata.zip`) を授業支援システムからダウンロードする。
(2) ファイルは圧縮 (zip) 形式なので，適当なフォルダに内容を展開（Windows 上ではダブルクリックして開き，中身を別フォルダにコピー）する。`city011`，`city012`，`city021`，`city022` の 4 つのフォルダがあり，各フォルダ内にそれぞれ 100 個ずつテキストファイルがある。
(3) 各ファイルには，`city011_001.txt` のような名前が付いている。この例で，最初の `city011` は「話者 01 の 1 回目発声」という意味である。したがって `city022` は「話者 02 の 2 回目発声」を意味する。
_（アンダーバー）の後の 3 桁数字は単語番号を表す。100 単語から成るので，001 から 100 までの番号が付いている。100 単語の発声内容は，すべて同じ順である。たとえば，先頭の単語（単語番号 001）は「AZABU（あざぶ）」である。
(4) ファイルはテキスト形式なので，Windows の「メモ帳」や UNIX (Cygwin) の cat コマンドで見ることができる。先頭の 3 行がヘッダ情報，4 行目から最後までがデータの本体。ヘッダ情報は，1 行目がファイル名から拡張子 `.txt` を除いたもの，2 行目が発声内容（音素の略式表示），3 行目が分析フレーム数。
フレーム数 50 の場合，ファイルの行数は 50 + 3（ヘッダ部）= 53 行となる。単語により長さが異なるので，ファイルの行数はそれぞれ異なる（同じ単語であっても発話時間が異なれば長さは異なる。）
(5) 各ファイルの 4 行目以降がデータ（音響特徴量ベクトル）を表す。それぞれの行に 15 個の浮動小数点数が書かれている。これは，線形予測（LPC）分析を用いて計算された**メルケプストラム**特徴量であり，1 フレームあたり 15 次から成る。

次に，認識実験の方法について説明する。

(1) 実験には，与えられた 4 つのデータセットのうち 2 つを用いる。1 つを**テンプレート（モデル）**，もう 1 つを**未知入力（認識対象）**データと考える。
たとえば，テンプレートとして `city011` データを，未知入力として `city012` データを用いる場合，同一話者による実験となる。テンプレートに `city011` を，未知入力に `city021` あるいは `city022` を用いると，別話者による実験となる。当然，同一話者の条件（特定話者）よりも別話者の条件のほうが認識は難しいので，認識性能（単語認識率）も別話者の場合が下がることが予想される。
(2) 実際に音声認識が行われる状況では，未知入力としてどんな単語が発声されるか分からないが，この課題では，100 単語のテンプレートに対して同じ 100 単語を順に入力していき，正しく認識されるかどうかを確認することで認識性能を調べる。
単語の発声内容はすべて同じであるから，たとえば未知入力の最初の単語（単語番号 001）をテンプレートの 100 単語とマッチングさせた結果，最初の単語に対して最も小さい累積距離が得られれば正解，そうでなければ不正解である。
(3) 100 単語の認識ができるようになったら，入力データセットを変えて実験してみる。同一話者については 2 つ，別話者については 4 つの組み合わせが考えられるので，可能ならばそれらのすべてについて調べてみること。

以下，プログラムの作り方に関するヒント：

(1) プログラミング言語は任意とする。慣れている環境を用いてよい。
(2) まず，テンプレートの 100 単語をすべて読み込んでおく。メモリが不足する場合は，1 単語ずつ読み込み，計算が終わったら次の単語，のようにしてもよい。
1 単語のデータ構造は `double data[frame][dimension]` のような 2 次元になる。
(3) 次に，認識対象単語（未知入力）データを同様のフォーマットで読み込む。次元数はどちらも 15 であるが，フレーム数は通常異なる。テンプレートと認識対象を混同しないように気をつけること。
(4) テンプレート中の単語 A（フレーム数 I）と未知入力単語 B（フレーム数 J）のマッチングを考える。A の i 番目フレームのデータ（15 次元）を $a_{i,k}$，B の j 番目フレームのデータを $b_{j,k}$ とすると，$a_{i,k}$ と $b_{j,k}$ のあいだの**局所距離**は，

$$d(i,j) = \sqrt{(a_{i,1}-b_{j,1})^2 + (a_{i,2}-b_{j,2})^2 + \cdots + (a_{i,15}-b_{j,15})^2}$$

で与えられる。
(5) これをすべてのフレーム相互間について計算した後，DP マッチングのアルゴリズムに従い，初期条件 g(0,0)，境界条件 g(i,0), g(0,j) およびその他の格子点における累積距離 g(i,j) を順次計算する。
(6) 初期条件：

$$g(0,0) = d(0,0)$$

(7) 境界条件：

$$i > 0 \text{ について } g(i,0) = g(i-1,0) + d(i,0)$$

$$j > 0 \text{ について } g(0,j) = g(0,j-1) + d(0,j)$$

(8) その他の格子点：

$$g(i,j) = \min \begin{bmatrix} g(i \quad , j-1 \quad) + d(i,j) \\ g(i-1 \quad , j-1 \quad) + 2d(i,j) \\ g(i-1 \quad , j \quad) + d(i,j) \end{bmatrix}$$

(9) 最終点までの累積距離 $g(I,J)$ が得られたら，その値を $(I+J)$ で割った値を A と B の**単語間距離**とする。
(10) テンプレートの 100 単語すべてについて単語間距離を計算し，その中で最小の距離を与えた単語を正解とする。便宜上，単語番号を出力すると正解か不正解かがすぐに分かる。正解単語と未知入力の単語番号が同じ場合は `o`，違う場合は `x` などと出力し，最後に `o` の数を数えればそれが単語正解率になる。（100 単語なので百分率を計算する必要はない）

ここまでが課題の内容（必修）。以下はオプショナルの考察事項（余力があればやってみる）。

(1) 認識が目的の場合，累積距離を得ることが重要であって，単語間のマッチングにおいてどのような経路（パス）を通ったかは陽に見えなくてもよい。しかし，最適経路がどのようになったかを見ることができるとなお面白い。累積距離の計算過程で，経路の情報（どちらから来た遷移が最小距離を与えたか）を順次保存しておき，最終点到達後に保存した情報を逆にたどることで，出発点に戻ることができる。これを適当な方法でグラフィック表示してみよ。
(2) DP の累積距離計算において，斜めの遷移の場合は局所距離を 2 倍している。これには理論的根拠があるが，この 2 倍を $\sqrt{2}$ 倍あるいは 1 倍に変えて実験せよ。認識性能に変化はあるか。
(3) DP マッチングにおける**整合窓**について調べて実装し，計算速度および認識性能にどのような変化があるかを調べよ。

- **レポート提出期限：2019 年 6 月 17 日 (月) 認識工学の講義開始前まで**
- **単位取得との関連：授業中に説明する通り**
- **レポート提出方法：授業支援システム上で提出**：必ず **1 つの PDF ファイル**にまとめること。
- **プログラムのソースコードと実行結果だけでなく，考察を必ず含めること。**