```
                Dstu(ii) = min(distance) + Dstu(ii);
            end
        end
        [val,pos] = min(Dstu);
        if val/size(mel,2) > =81
            fprintf('测试者不是系统内人\n')
        else
            fprintf('测试者为 SX%d\n',pos)
        end
    end
end
```

11.2　基于高斯混合模型（GMM）的说话人识别实验

11.2.1　实验目的

1）掌握高斯混合模型（GMM）的定义和参数估计算法。

2）掌握基于 GMM 的说话人识别基本过程。

3）应用 MATLAB 实现基于 GMM 的说话人识别。

11.2.2　实验原理

1. 高斯混合模型（GMM）的定义和参数估计算法

高斯混和模型（Gaussian Mixture Model，GMM）可以看作一种状态数为 1 的连续分布隐马尔可夫模型。一个 M 阶混合高斯模型的概率密度函数是由 M 个高斯概率密度函数加权求和得到的，所示如下：

$$P(\boldsymbol{X}/\boldsymbol{\lambda}) = \sum_{i=1}^{M} \boldsymbol{w}_i b_i(\boldsymbol{X}) \tag{11-8}$$

其中，$\boldsymbol{X}$ 是 D 维随机向量，为说话人识别算法提取的特征矢量；$w_i, i = 1, \cdots, M$ 是混合权重，满足 $\sum_{i=1}^{M} w_i = 1$。每个子分布 $\boldsymbol{b}_i(\boldsymbol{X}_t), i = 1, \cdots, M$ 是 D 维的联合高斯概率分布，可表示为

$$\boldsymbol{b}_i(\boldsymbol{X}) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}_i|^{1/2}} \exp\left\{ -\frac{1}{2} (\boldsymbol{X} - \boldsymbol{\mu}_i)' \Sigma_i^{-1} (\boldsymbol{X} - \boldsymbol{\mu}_i) \right\} \tag{11-9}$$

其中，$\boldsymbol{\mu}_i$ 是均值向量；$\boldsymbol{\Sigma}_i$ 是协方差矩阵。完整的混合高斯模型由参数均值向量，协方差矩阵和混合权重组成，表示为：$\boldsymbol{\lambda} = \{\boldsymbol{w}_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}, i = 1, \cdots, M$。

GMM 模型的参数估计算法是给定一组训练数据，依据某种准则确定模型的参数 λ 的过程。最常用的参数估计方法是基于最大似然（Maximum Likelihood）准则的估计。在具体实现时，通常采用期望值最大（Expectation Maximization，EM）算法估计参数 $\boldsymbol{\lambda}$，即，采用 EM 算法估计出一个新的参数 $\hat{\boldsymbol{\lambda}}$，使得新的模型参数下的似然度为 $P(\boldsymbol{X}/\hat{\boldsymbol{\lambda}}) \geqslant P(\boldsymbol{X}/\boldsymbol{\lambda})$。新的模型参数再作为当前参数进行新一轮的重新估计，这样迭代运算直到模型收敛，从而保证了模型似然度的单调递增。具体算法原理与步骤参见教材的相关章节。每次迭代中，三组参数的

重估公式如下：

混合权值的重估公式

$$\boldsymbol{w}_i = \frac{1}{T}\sum_{t=1}^{T} P(i/\boldsymbol{X}_t, \boldsymbol{\lambda}) \tag{11-10}$$

均值的重估公式

$$\boldsymbol{\mu}_i = \frac{\sum_{t=1}^{T} P(i/\boldsymbol{X}_t, \boldsymbol{\lambda})\boldsymbol{X}_t}{\sum_{t=1}^{T} P(i/\boldsymbol{X}_t, \boldsymbol{\lambda})} \tag{11-11}$$

方差的重估公式

$$\boldsymbol{\sigma}_i^2 = \frac{\sum_{t=1}^{T} P(i/\boldsymbol{X}_t, \boldsymbol{\lambda})\ (\boldsymbol{X}_t - \boldsymbol{\mu}_i)^2}{\sum_{t=1}^{T} P(i/\boldsymbol{X}_t, \boldsymbol{\lambda})} \tag{11-12}$$

其中，分量 i 的后验概率为

$$P(i/\boldsymbol{X}_t, \boldsymbol{\lambda}) = \frac{\boldsymbol{w}_i \boldsymbol{b}_i(\boldsymbol{X}_t)}{\sum_{k=1}^{M} \boldsymbol{w}_k \boldsymbol{b}_k(\boldsymbol{X}_t)} \tag{11-13}$$

2. 基于 GMM 的说话人识别过程

图 11-4 为基于 GMM 的说话人识别过程示意图，主要包括两个过程：①对每个说话人 $n=1,\cdots,N$ 的训练语音集，提取特征参数，基于最大似然准则，并且通过上一部分介绍的 EM 算法，建立与该说话人对应的高斯混合模型 $\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2,\cdots,\boldsymbol{\lambda}_N$；②对于待识别说话人的语音，首先提取特征参数，而后计算其关于训练后得到的每一个说话人模型 $\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2,\cdots,\boldsymbol{\lambda}_N$ 的似然值 $p(X/\boldsymbol{\lambda}_n)$，将其中的最大似然值对应的序号作为说话人识别结果：$n^* = \arg\max_n p(X/\boldsymbol{\lambda}_n)$。

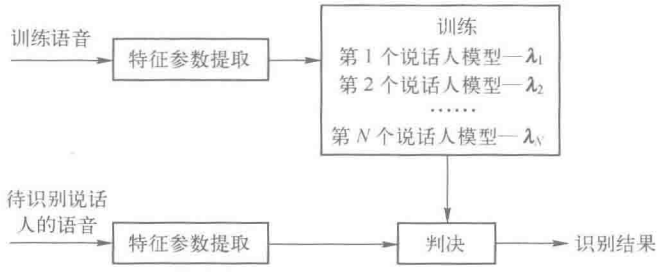

图 11-4 基于 GMM 的说话人识别的过程框图

在实际应用中，需要注意如下两个问题：

1）GMM 模型的高斯分量的个数 M 和模型的初始参数必须首先确定。其中，最优高斯分量 M 的大小，很难从理论上推导出来，可以根据不同的识别系统，由实验确定。一般，M 取值可以是 4、8、16 等，在本实验中，取 $M=16$。对于模型参数初始化问题，首先采用聚类的方法将特征矢量归为与混和数相等的各个类中，然后分别计算各个类的方差和均值，作为初始矩阵和均值，权值是各个类中所包含的特征矢量的个数占总的特征矢量的百分比。建立的 GMM 中，方差矩阵可以为全矩阵，也可以为对角矩阵，这里采用对角矩阵。

2）在实际应用中，往往得不到大量充分的训练数据对模型参数进行训练。由于训练数据的不充分，GMM 模型的协方差矩阵的一些分量可能会很小，这些很小的值对模型参数的

似然度函数影响很大，严重影响系统的性能。为了避免小的值对系统性能的影响，在 EM 算法的迭代计算中，对协方差的值设置一个门限值，在训练过程中令协方差的值不小于设定的门限值，否则用设置的门限值代替。门限值设置可通过观察协方差矩阵来定。

11.2.3　实验步骤和实验结果

1. 关键函数说明

GMM 参数估计函数：gmm_emm

功能：基于最大似然准则的 EM 参数估计算法

调用格式：

```
[mix,post,errlog] = gmm_em(mix,x,emiter)
```

参数说明：输入参数 mix 为经过初始化的 mix 结构体，其中保存着 GMM 的模型参数；x 为用于训练的语音信号的 MFCC 特征序列，emiter 为 EM 算法运行的最大迭代次数。输出参数 mix 为训练完成后的 mix 结构体，post 为后验概率。

程序清单：

```
function[mix,post] = gmm_em(mix,x,emiter)
[dim,data_sz] = size(x');
init_covars = mix.covars;'
MIN_COVAR = 0.001;
  for cnt = 1:emiter
    % ---E step:计算充分统计量---
    [post,act] = calcpost(mix,x);
    prob = act * (mix.priors)';
    errlog(cnt) = -sum(log(prob));
    % ---M step:重估三组参数---
    new_pr = sum(post,1);
    new_c = post' * x;
    mix.priors = new_pr./data_sz;                          %重估权重
    mix.centres = new_c./(new_pr' * ones(1,dim) + eps);%重估均值矢量
    switch mix.covar_type                                  %重估协方差矩阵
    case 'diag'                                            %当协方差矩阵为对角阵时的重估过程
      for j = 1:mix.ncentres
        diffs = x - (ones(data_sz,1) * mix.centres(j,:));
        mix.covars(j,:) = sum((diffs.* diffs)…
                    .* (post(:,j) * ones(1,dim)),1)./new_pr(j);
        if min(mix.covars(j,:)) < MIN_COVAR
          mix.covars(j,:) = init_covars(j,:);
        end
      end
    case 'full'                                            %当协方差矩阵为一般矩阵时的重估过程
      for j = 1:mix.ncentres
```

```
            diffs = x - (ones(data_sz,1) * mix.centres(j,:));
            diffs = diffs. * (sqrt(post(:,j)) * ones(1,dim));
            mix.covars(:,:,j) = (diffs' * diffs)/(new_pr(j) + eps);
            if min(svd(mix.covars(:,:,j))) < MIN_COVAR
              a = svd(mix.covars(:,:,j));
              mix.covars(:,:,j) = init_covars(:,:,j);
            end
          end % end of "for j = 1:mix.ncentres"
        otherwise
          error(['Unknown covariance type ',mix.covar_type]);
        end
    end % end of "for cnt = 1:emiter"
```

2. 实验步骤

本实验分为训练过程和识别过程，具体步骤如下。

（1）训练过程（运行 train.m 文件）

1）本实验的语音来自 6 个说话人，每人有 5 段语音，保存在 tra_data.mat 中。首先载入该文件，得到一个二维结构体 tdata{i}{j}，i = 1，…，6；j = 1，…，5，其表示第 i 个说话人的第 j 段语音。以 tdata{2}{3}为例，将其保存在 speech 变量中。

2）对 speech 进行预处理和特征提取。预处理主要包括预加重、分帧、加窗等过程，而特征提取则是调用 melcepst 函数，提取 MFCC 特征参数。

3）调用 gmm_init 函数，设定 GMM 中各参数 $\boldsymbol{\lambda} = \{w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}, i = 1, \cdots, M$ 的初始值，该函数中采用的方法是 k 均值聚类法。注意，变量 kiter 控制 k 均值聚类的最大迭代次数。

4）调用 gmm_em 函数，用 EM 算法对 GMM 的各参数进行重估更新，经过 emiter 次迭代，得到第 2 个说话人对应的模型——$\boldsymbol{\lambda}_2$。

5）用同样的方式得到每个说话人对应的模型 $\boldsymbol{\lambda}_1, \cdots, \boldsymbol{\lambda}_6$，分别保存 speaker{1}，…，speaker{6}中，最后将 speaker 结构体存入到 speaker.mat 文件中。

训练结果如图 11-5 所示。

图 11-5 训练结果

（2）识别过程（运行 recog. m 文件）

1）本实验用于识别的语音保存在 rec_data. mat 中，首先载入该文件，得到一个二维结构体 rdata{i}{j}，i = 1，…，6；j = 1，2，即每个说话人有两条待测试语音。如 rdata{4}{1}表示该语音是由第 4 个说话人发出的（识别之前不知道该结果）。

2）载入训练好的说话人模型 $\boldsymbol{\lambda}_1, \cdots, \boldsymbol{\lambda}_6$：load speaker. mat。

3）对待识别语音，进行预处理，并且调用 melcepst 函数，进行特征提取。

4）对于待识别语音的特征矢量 $\boldsymbol{X}$，计算其关于各说话人模型的似然值，以 $\boldsymbol{\lambda}_1$ 为例，计算过程为

$$P(\boldsymbol{X}/\boldsymbol{\lambda}) = \sum_{i=1}^{M} w_i b_i(\boldsymbol{X}) \tag{11-14}$$

其中，w_i 和 $b_i(\boldsymbol{X})$ 中的参数 $\boldsymbol{\mu}_i$、$\boldsymbol{\sigma}_i^2$ 由训练完成时估计得到（程序中分别为 speaker{1}. pai，speaker{1}. mu 和 speaker{1}. sigma）。

5）选择最大似然值对应的序号作为说话人识别结果。

需要说明和注意的是，本实验中，混合成分数 ncentres，K 均值聚类的最大迭代数 kiter，EM 算法最大迭代数 emiter 都为可调节参数，可以设定不同的值，比较不同条件下的说话人性能。

识别结果如图 11-6 所示。

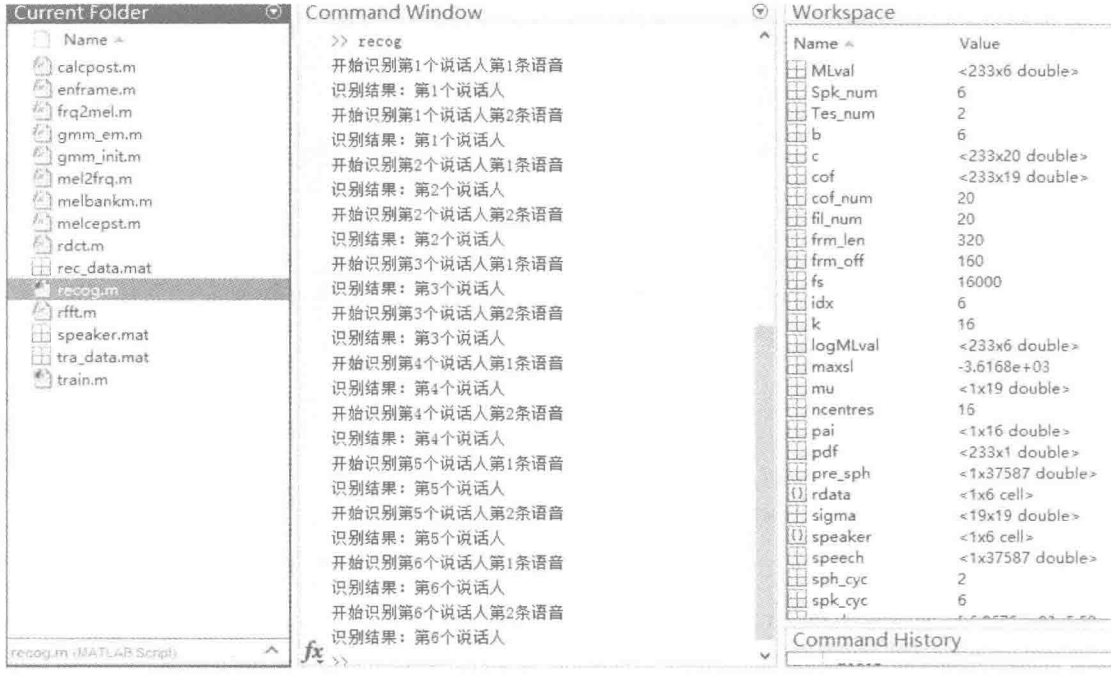

图 11-6　识别结果

11.2.4　思考题

1）改变 GMM 的混合成分数，编程比较不同状态数条件对识别结果的影响。

2）改变训练语音的段数，编程比较不同训练数据量对识别结果的影响。

11.2.5 参考例程

```
% 基于 GMM 的说话人识别
% --识别---
clear all;
load rec_data. mat;                                  % 载入待识别语音
load speaker. mat;                                   % 载入训练好的模型
Spk_num = 6;                                         % 说话人个数
Tes_num = 2;                                         % 每个说话人待识别的语音数目
fs = 16000;                                          % 采样频率
ncentres = 16;                                       % 混合成分数目

for spk_cyc = 1:Spk_num                              % 遍历说话人
  for sph_cyc = 1:Tes_num                            % 遍历语音
    fprintf('开始识别第%i 个说话人第%i 条语音\n',spk_cyc,sph_cyc);
    speech = rdata{spk_cyc}{sph_cyc};
    % ---预处理,特征提取--
    pre_sph = filter([1 -0.97],1,speech);
    win_type = 'M';                                  % 汉明窗
    cof_num = 20;                                    % 倒谱系数个数
    frm_len = fs * 0.02;                             % 帧长:20ms
    fil_num = 20;                                    % 滤波器组个数
    frm_off = fs * 0.01;                             % 帧移:10ms
    c = melcepst(pre_sph,fs,win_type,cof_num,fil_num,frm_len,frm_off);
    cof = c(:,1:end-1);                              % N * D 维矢量
    % ----识别---
    MLval = zeros(size(cof,1),Spk_num);
    for b = 1:Spk_num                                % 说话人循环
    pai = speaker{b}. pai;
    for k = 1:ncentres
      mu = speaker{b}. mu(k,:);
      sigma = speaker{b}. sigma(:,:,k);
      pdf = mvnpdf(cof,mu,sigma);
      MLval(:,b) = MLval(:,b) + pdf * pai(k);        % 计算似然值
    end
   end
   logMLval = log((MLval) + eps);
   sumlog = sum(logMLval,1);
   [maxsl,idx] = max(sumlog);                        % 判决,将最大似然值对应的序号 idx 作为
识别结果
   fprintf('识别结果:第%i 个说话人\n',idx);
  end
end
```