

**实验报告**

**实 验（三）**

题 目 命令词识别实验报告

专 业 人工智能

学　　 号 1190201215

班　　 级 1903602

学 生 冯开来

指 导 教 师 郑铁然

实 验 地 点 格物207

实 验 日 期 2021.12.24

**计算机科学与技术学院**

# 

# 一、 设计命令词识别任务

## 1.1 描述所设计的命令词识别任务

1. 共录制5个命令词，每组10个，共50个wav文件。存储在文件夹“语料”中。
2. 进行端点检测，去除50个音频的静音部分。替换原来的音频文件。
3. 利用HTK中的HCopy工具提取出i\_j.wav文件的MFCC特征，存放在i\_j.mfc文件中，同样放在文件夹“语料”中。
4. 编写代码提取出MFCC特征值，一个MFCC为一个帧数×维度的矩阵，共50个MFCC矩阵，构成一个5×10的矩阵。
5. 对50个MFCC分别使用DTW模型和HMM模型进行匹配。其中，DTW取每个音频的第1个为训练模板，剩余9个为测试样例，即共有5个训练样本，45个测试样本；HMM模型取每个音频的前3个为训练模板，后7个为测试样例，即共15个训练样本，35个测试样本。
6. 分别对上述两种模型的测试样本进行匹配，输出每个测试样例的识别结果（标签表示），并计算两种模型的正确率。

## 1.2 列出词表

打开电脑

关闭电脑

开始移动

保持静止

语音识别

## 1.3 介绍语料采集方法和规模，

采集方法为使用CoolEdit Pro进行录制；

录制规模为5个命令词，每个录制10遍共50个。

# 二、特征提取

## 2.1 详细描述你所采用的特征和提取算法

1. 采用特征：

利用HCopy，设置参数帧长25ms，帧移10ms。计算出的39维MFCC特征。每个特征的值为4Bytes，存放在帧数nframes×特征数ndim的数组中。39维特征包括13维的MFCC系数，13维的一阶差分参数，13位的二阶差分参数。

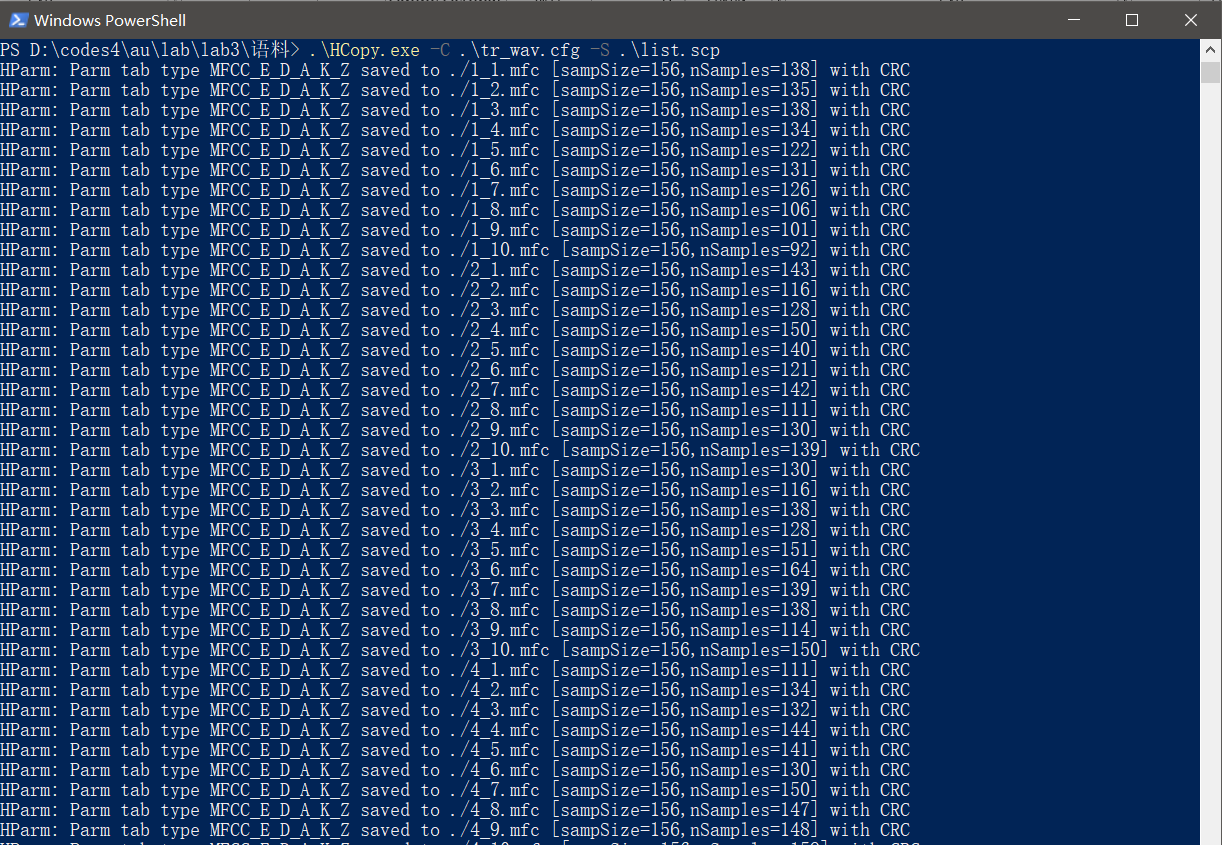
1. 提取算法：

利用HTK中的HCopy提取，存储为.mfc文件。首先进行预加重，然后进行分帧，加窗，然后进行快速傅里叶变换，将它转换为频域上的能量分布来观察；将能量谱通过一组Mel尺度的三角形滤波器组，对频谱进行平滑化，并消除谐波的作用，突显原先语音的共振峰；计算每个滤波器输出的对数能量，经离散余弦变换（DCT）得到MFCC系数；然后计算对数能量；最后提取动态差分参数（包括一阶差分和二阶差分）。

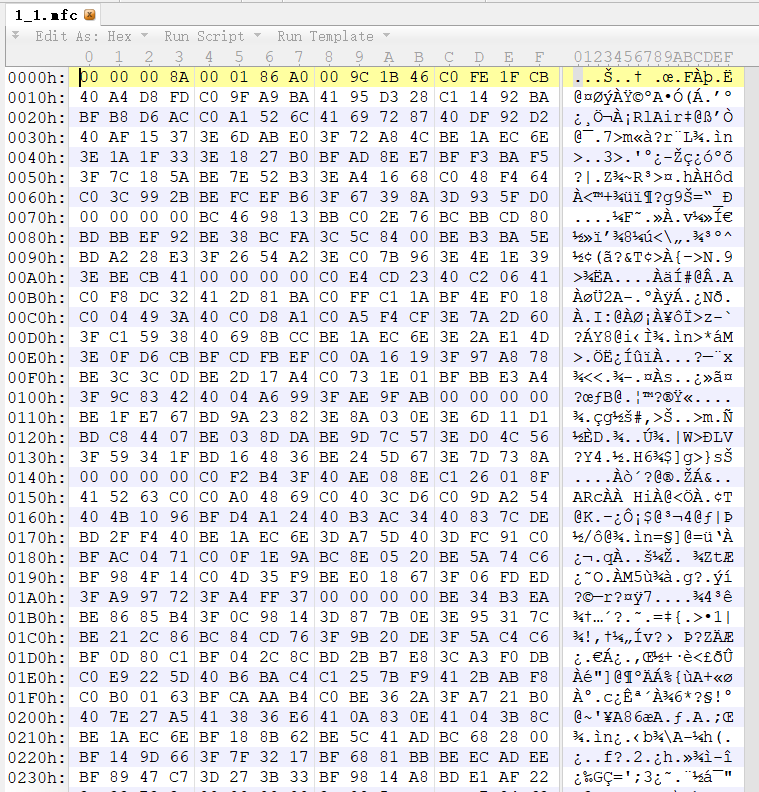
2.2 给出特征提取部分运行结果的截图

.\HCopy.exe -C .\tr\_wav.cfg -S .\list.scp

wav.cfg为配置文件名。list.scp为存放wav文件和提出对应特征文件的名



2.3 给出特征文件内容的截图（二进制打开或逐维读出到文本文件）



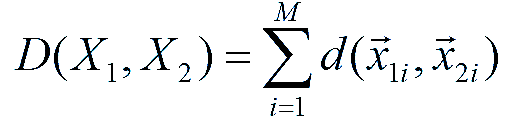
（1\_1.mfc文件二进制打开）

# 三、 基于DTW的命令词识别

## 3.1 介绍你所设计DTW算法，标明所采用的开发工具

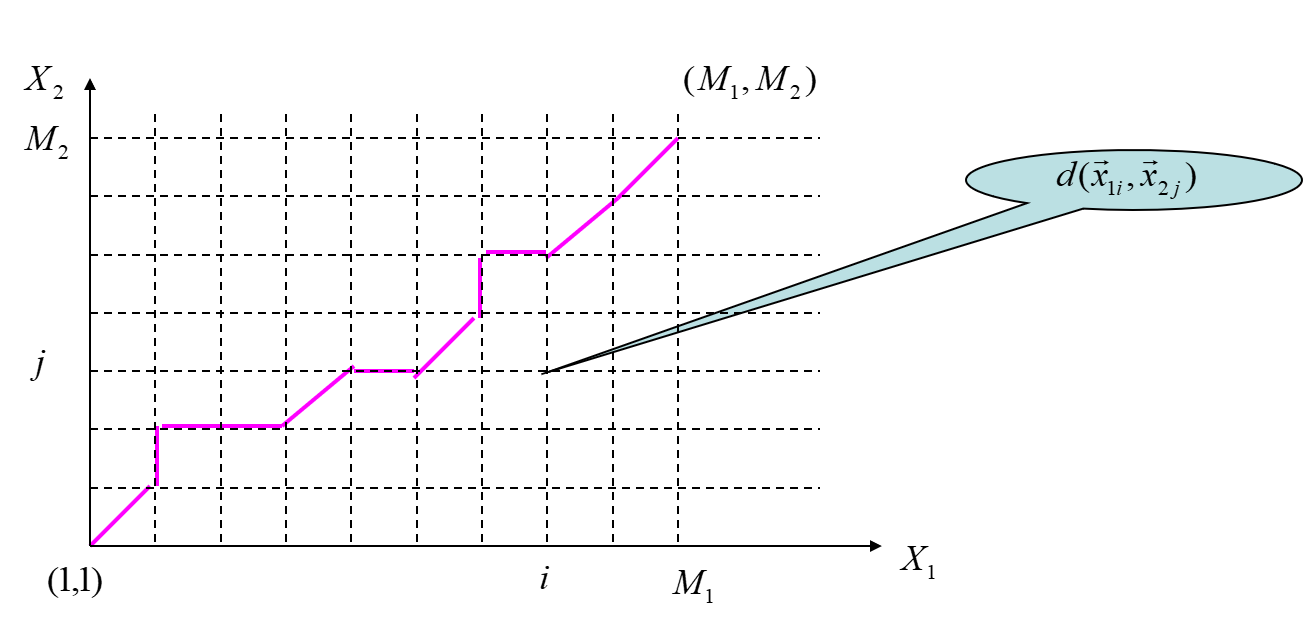
简单的来说就是计算测试样本和训练样本哪个更相似，相似度通过两个MFCC特征距离来计算。不过MFCC特征是一个矩阵，计算距离可以通过每一帧的欧式距离的和来表示。不过有一个问题就是每两个MFCC特征的帧数不一定一样，所以在算每一帧的欧式距离的时候肯定有些帧就空着没法算。所以就引入了DTW算法。

这是一个动态规划算法。对两个不同维数的语音矩阵model，test进行匹配。Model，test的每一维都是一个向量，是语音一帧的MFCC特征值。两个矩阵model，test的帧数分别为M1，M2，则距离可以表示为：

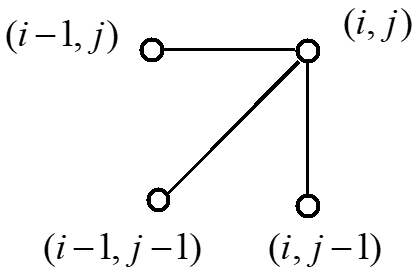


那么，进行如下匹配：

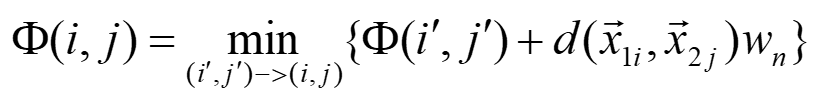
1. 每一条从（1，1）到（M1，M2）的路径都有一个累加距离称为路径的代价。
2. 每一条路径代表一种对齐情况。



1. 代价最小的路径就是所求的对准路径
2. 如何找到最小路径，首先对搜索路径进行如下限制，（i，j）仅能通过（i-1，j）或（i-1，j-1）或（i，j-1）到达。



1. 因为有了这个限制，全局最优问题转换为局部最优问题（i，j）的代价可以通过更低一层的代价×系数累加得到。

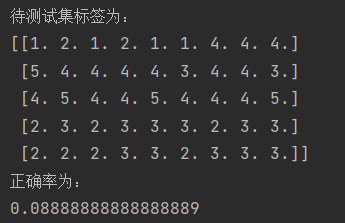


1. 最后可以从（1，1）开始计算，定义加权系数（与局部路径有关）：
2. 递推求累计距离，得到的（M1，M2）的代价就是两个矩阵之间的距离。
3. 代码如下：

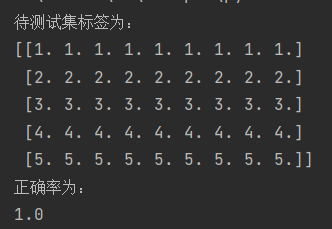
**def** DTW(x1, x2):  
 x1 = np.array(x1)  
 x2 = np.array(x2)  
 row = len(x1)  
 col = len(x2)  
 ndim = len(x1[0])  
 D = np.zeros((row, col))  
 *# 初始化（i，j）的代价* **for** i **in** range(row):  
 **for** j **in** range(col):  
 D[i][j] = 0  
 D[i][j] = np.linalg.norm(x1[i] - x2[j]) *# 欧式距离  
 # for k in range(ndim):  
 # D[i][j] += abs(x1[i][k] - x2[j][k]) # 每一个差值的 绝对值，最后求和* **for** i **in** range(1, row): *# 计算每一列代价路径的累计距离* D[i][0] += D[i-1][0]  
 **for** j **in** range(1, col): *# 计算每一行代价路径的累计距离* D[0][j] += D[0][j-1]  
 **for** i **in** range(1, row): *# 开始搜索最短代价路径* **for** j **in** range(1, col):  
 D[i][j] = min(D[i-1][j]+D[i][j], D[i-1][j-1]+2\*D[i][j], D[i][j-1]+D[i][j])  
 **return** D[row-1][col-1] / (row + col -2) *# 结果用系数和归正  
 # return D[row-1][col-1]*

3.2 **正确率**：100%

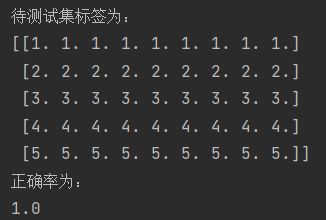
在一开始向量距离使用欧式距离计算的时候，正确率只有8%，非常低，我也不知道为什么。



后来计算向量距离我使用每个点的差的绝对值并且累加，这时候计算出来正确率有所提升，直接全部正确



所以我后来又重新看了一下计算欧式距离的代码，发现有的算出来是负数没有取绝对值（欧氏距离算错了）。经过修改后，最后计算正确率结果为100%



# 五、 总结

## 5.1 请总结本次实验的收获

1. 学会了如何使用HCopy进行提取每个音频的MFCC特征文件
2. 掌握了MFCC特征的含义
3. 学会了DTW算法和在语音识别中的应用。

## 5.2 请给出对本次实验内容的建议

HMM有点难，很难，太难了