实验一 基于 DTW 的语音数字识别

学号：2019101404　姓名：马正一

1 任务描述

（1） 自行构建英文数字语音数据集。
（2） 基于数字语音数据集，编写代码，使用 DTW 算法完成语音识别，对输入的一段音频进行分类，输出语音中的数字，如“2”、“10”。

2 实验环境

操作系统使用 MacOS，Python=3.6，python-speech-features，pyaudio。

3 实验方案

3.1 构建数据集

我们使用音频处理工具 Pratt，录制了英文数字 1-10 的录音文件，存为 wav 文件到本地。对于每个数字，我们录制五个样本，其中一个音频作为模板，另外四个模板作为测试集。每个样本的时长为 2-3 秒。

3.2 端点检测

由于我们的数据集为人工录制，每个录制样本在真正的数字音段前后都会有一些噪声段，直接用原始数据执行接下来的语音识别算法效果可能会相对较差。因此，我们在进行语音识别前先进行端点检测，取出语音中的静音部分。

在端点检测模块，我们使用双门限法进行端点检测。算法一共分为三步，每一步对应一个阈值，包括两个短时能量阈值和一个过零率阈值。第一步是取一个较高的短时能量作为阈值 MH，利用这个阈值，我们就可以先分出语音中的浊音部分。第二步是取一个较低的能量阈值 ML，利用这个阈值，我们可以从 A1，A2，向两

端进行搜索，将较低能量段的语音部分也加入到语音段，进一步扩大语音段范围。第三步是利用短时过零率，短时过零率的阈值为 Zs。由于语音的两端部分是辅音（也就是清音部分），也是语音中的一部分，但是辅音的能量与静音部分的能量一样低，但是过零率比静音部分高出很多。为了区分开二者，将利用短时能量区分完的语音段继续向两端进行搜索，短时过零率大于 3 倍 Zs 的部分，则认为是语音的清音部分。将该部分加入语言段，就是求得的语音段。

我们在项目的 *endpointDetection_RecordedVoice.py* 文件中编写了端点检测的代码。运行该 py 文件可将原始的 1-10 数字共 50 个 wav 文件处理为端点检测后的 50 个 wav 文件。

3.3 MFCC 特征提取

完成端点检测算法之后，我们接下来对处理好的音频文件进行特征提取，以供接下来的 DTW 算法使用。我们在本实验中采用的是利用 python 的 python-speech-features 库计算出的 39 维 MFCC 特征，包括 12 维 dct 系数，能量特征，一阶差分和二阶差分等。具体来说，算法首先进行预加重，然后对语音文件进行分帧，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）

我们在项目的 *VoiceRecog.py* 文件中编写了 MFCC 特征提取代码。在 *VoiceRecog.py* 中的 *extract_MFCC* 函数我们会对输入的每个文件，返回 39 维的 Numpy 特征数组。

3.4 DTW 算法

提取 MFCC 特征之后，我们开始使用 DTW 算法进行语音识别。DTW 本质上是一个简单的动态规划算法，是用来计算两个维数不同的向量之间的相似度的问题，即计算向量 M1 和 M2 的最短距离。

算法步骤如下：

1. 初始化：

$$i = j = 1,\ \Phi(1,1) = d(\vec{x}_{11}, \vec{x}_{21})$$

$$\Phi(i,j) = \begin{cases} 0 & \text{当}(i,j) \in \text{Reg} \\ huge & \text{当}(i,j) \notin \text{Reg} \end{cases}$$

其中约束区域 Reg 可以假定是这样一个平行四边形，它有两个顶点位于(1,1)和(M_1,M_2)，相邻两条边的斜率分别为 2 和 1/2。

2. 递推求累计距离 并记录回溯信息:

$$\Phi(i,j)=\min\{\Phi(i-1,j)+d(\vec{x}_{1i},\vec{x}_{2j})\cdot W_n(1);\Phi(i-1,j-1)+d(\vec{x}_{1i},\vec{x}_{2j})\cdot W_n(2);$$

$$\Phi(i,j-1)+d(\vec{x}_{1i},\vec{x}_{2j})\cdot W_n(3)\}$$

$$i=2,3,\cdots,M_1;j=2,3,\cdots,M_2;(i,j)\in \mathrm{Reg}$$

一般取距离加权值为:

$$W_n(1)=W_n(3)=1 \qquad W_n(2)=2$$

3. 计算出的值就是 m1 和 m2 之间的距离。

具体来说，我们在本实验中会将每个数字随机选取一个样本作为模板文件。对剩下的 40 个测试语音文件，我们对每个测试样本使用 DTW 算法计算与模板音频之间的距离，选取距离最短的模板作为其标签输出。运行 *VoiceRecog.py* 即可完成特征提取、语音识别、测试实验。

4 运行手册

本项目已上传至 github 并编写 Readme 说明手册，链接为：Link

在实验开始前，用户需使用 Anaconda 或 pip 安装实验所需的 python 环境。

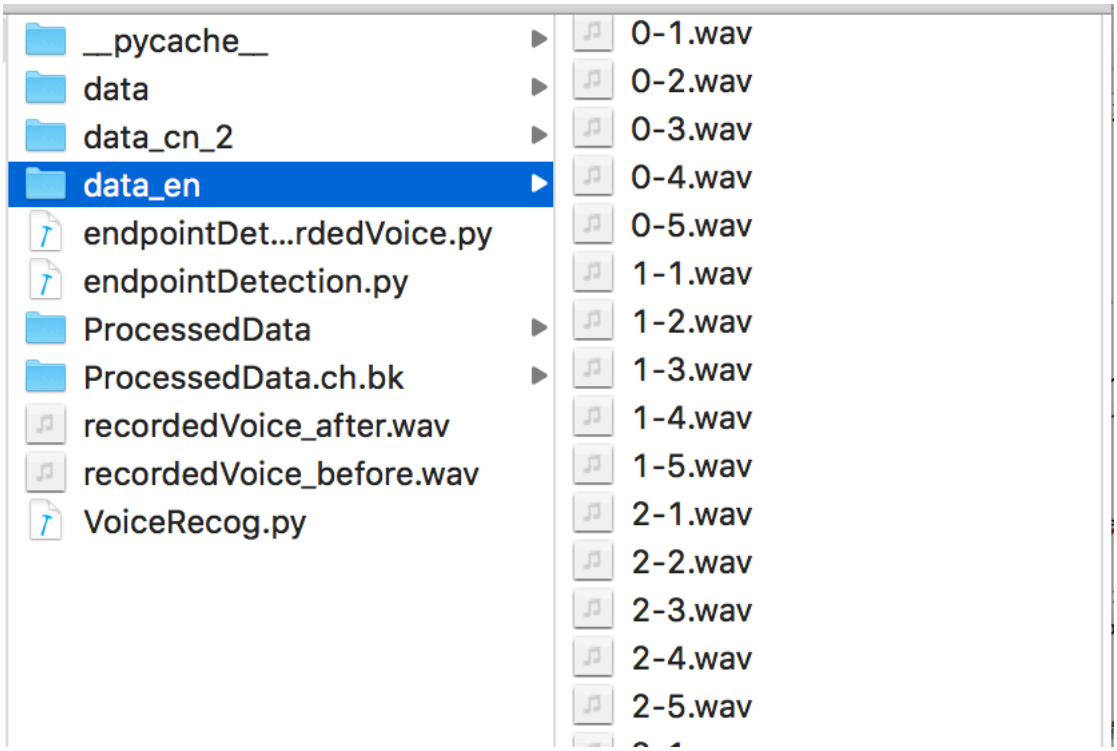
```
conda create -n dtw -c anaconda python=3.6 numpy tqdm pyaudio scipy #也可以使用 pip
 conda activate dtw
 pip install python_speech_features
```

之后，运行以下命令完成实验并查看实验结果

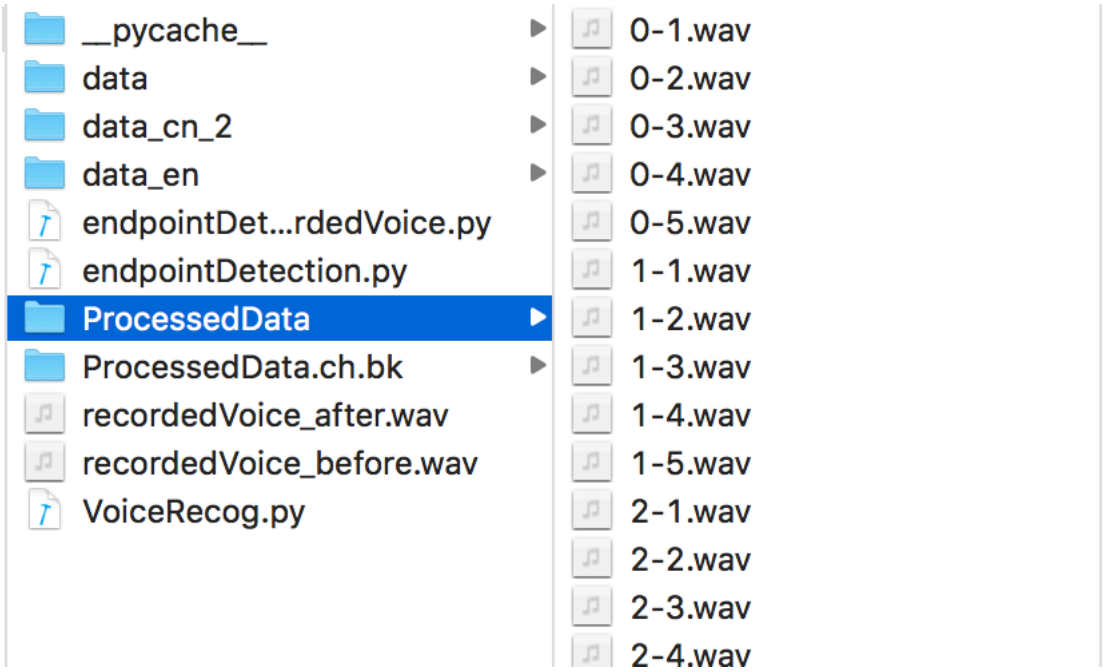
```
  git clone https://github.com/zhengyima/DTW_Digital_Voice_Recognition.git DTW_DVR
  cd DTW_DVR
  mkdir ProcessedData # 创建端点处理后数据目录
  python endpointDetection_RecordedVoice.py # 端点检测，默认使用英文数据
  python VoiceRecog.py
```

5 实验结果及运行截图

本实验在实验者本人制作的英文数据集下，对测试的 40 个样本可正确识别其中 38 个样本，准确率达到 95%。



原始数据文件



端点检测后的数据文件

```
5-2,pred:1,true:5
5-3,pred:1,true:5
6-0,pred:6,true:6
6-1,pred:6,true:6
6-2,pred:6,true:6
6-3,pred:6,true:6
7-0,pred:7,true:7
7-1,pred:7,true:7
7-2,pred:7,true:7
7-3,pred:7,true:7
8-0,pred:8,true:8
8-1,pred:8,true:8
8-2,pred:8,true:8
8-3,pred:8,true:8
9-0,pred:9,true:9
9-1,pred:9,true:9
9-2,pred:9,true:9
9-3,pred:9,true:9
acc:0.950000
start recording realtime...
recording finished!
[8, 50, 102, 155]
        8

(slp) MazydeMacBook-Pro-2:lab1 lmerengues$
```

运行 *VoiceRecog.py* 进行实验，准确率达到 95%