浙江大学软件学院 语音识别识别项目实验报告

一：

项目的源代码是根据github上一个类似的项目改写而成

<https://github.com/crouchred/speaker-recognition-py3>

项目的主体流程还是按照 群里提供的资料 ：即 mfcc + gmm 模式

改写的地方 主要有：

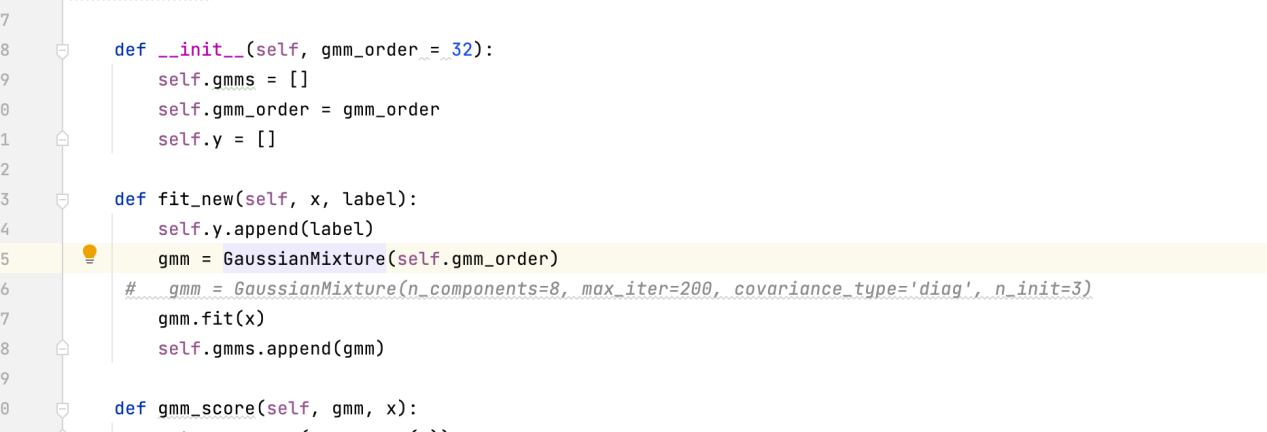
输入输出的改写，要适配成 符合格式的txt作为输入和输出，这一部分主要写在test.py这个文件下了，都是写成了辅助函数的形式

mfcc 初始化参数的改动：初始化参数用的就是用的 评测基准系统.pdf中给的初始化参数



gmm相关参数没有选择用提供的，选择的是指定了一个参数，其他自己运算生成

当然，也试过 pdf中指定的参数生成的模型，不过效果不太好



项目的本质就是一个 语音标签二分类项目

因为标签只有两个，模型会给两种标签分别打分，得分高的就是预测的标签

后续因为其实可以设置参数的地方很多，生成的模型就会有多个，因为是二分类，所以理论上只要有三个预测概率比较接近的模型（概率大于50%），就可以通过投票的机制得出一个理论上正确率更高的结果。这里是概率论的相关知识，例如三个模型的准确率均接近 q，且q>0.5，那么 采用 投票法 ，选取三个中标签数最多的那个作为最终标签，因为3是奇数，所以一定会有一个结果，这个结果的正确率为：

q\*q\*q+3\*(1-q)\*q\*q ,因为q大于0.5，从数学上可以得出这个正确率是大于q的

这个方法作为一个辅助函数写在了 test.py 里的threeToOne()函数里面，前提是当前已经有了三个准确率差不多的预测结果，然后会生成一个准确率更高的预测结果

源代码最后会把整个项目传到 github上对应 的我自己名字的文件夹下

由于项目中的venv（环境之类的文件夹），model模型文件夹过大，超过了github100m的上传限制，所以就不上传了，但是应该不影响，model可以自己生成，venv里面就只有pip和python3之类的东西

二．

实验过程：

跑python脚本都是用命令行直接跑的

例如：

./speaker-recognition.py -t enroll -i "./train/flac ./dev/flac" -m model.out

代表了 train+ dev 一起训练 当然要确保 train和dev文件夹的存在，而且里面放的就是音频文件

./speaker-recognition.py -t enroll -i "./train" -m model.out 训练也可以只训练这一个文件夹

训练之后会产生模型，这个过程有可能会比较长，比如不设定gmm参数的话我电脑跑了20分钟才跑出模型

然后就可以预测了

也是跑脚本

输出的文件名.txt 由 test.py 里面的reshelp()辅助函数中指定

./speaker-recognition.py -t predict -i "./eval2/flac/\*.flac" -m model1.out

./speaker-recognition.py -t predict -i "./eval2/flac/\*.flac" -m model2.out

./speaker-recognition.py -t predict -i "./eval2/flac/\*.flac" -m model3.out

./speaker-recognition.py -t predict -i "./eval2/flac/\*.flac" -m model4.out

这里的eval2可以改成对应的eval1之类的 ，模型的话每个有100多m，会单独建一个文件夹上传

目前最后是用的 model 2，model3，model4，生成三个结果，然后用那个test.py里面的 threeToOne()辅助函数生成第四个结果，生成的时候 记得手动修改 test.py 里面的reshelp()函数里的最终文件名

因为准确率差别不大，可以简化过程

跑一个模型就行了

./speaker-recognition.py -t enroll -i "./train/flac ./dev/flac" -m model.out

这一行来生成模型 ，前提是 对应的文件夹存在

./speaker-recognition.py -t predict -i "./eval1/flac/\*.flac" -m model.out

这一行去预测