Project 1：语音端点检测

522030910109 刘翰文

1. 基于线性分类器和语音短时能量的简单语音端点检测算法

1.1. 数据预处理及特征提取

任务 1 要求利用语音的短时信号特征（短时能量，过零率，短时频谱，以及基频等）使用线性分类器进行简单的语音端点检测。本次任务在对过零率、短时能量以及二者的线性组合中依据最终效果选择了短时能量作为语音的特征，用语音的短时能量进行语音端点检测。语音的短时能量定义如式1:

$$E = \sum_{n=0}^{N-1} s^2(n) \tag{1}$$

短时能量在语音识别中有很多的作用，短时能量可以区分清音和浊音，因为浊音的能量要比清音的大得多；也能区分语音短和非语音段，因为语音段的能量要大于非语音段的能量；还能对声母和韵母进行分界以及对连字的分界等。其在人耳造成的感觉便如上述作用一般，体现在清浊音，语音非语音上等。

过零率是在声音信号的每一帧中，声音信号的采样值通过零点的次数。过零率的定义如式2:

$$Z = \frac{1}{2}\{\sum_{n=0}^{N-1} |sgn[s(n)] - sgn[s(n-1)]|\} \tag{2}$$

过零率在轻音、噪音与浊音的识别上发挥着重要的作用，轻音与噪音的过零率一般要大于浊音的过零率。

对于提供的一整段语音信号，我首先对每段信号都进行了分帧的预处理，该处理可以将一段长语音序列分为许多有重叠的短时语音序列，这是进行语音端点检测必要的一步。因为一段较长的语音信号是不稳定的，它的特征和参数随时间变化程度较大，是一个非平稳态的过程，而分帧的操作可以将语音序列切成一段段短时语音信号，而这些信号可以认为特征基本不变，具有短时平稳性。所以可以按帧为单位，从帧的角度判断该帧是否为语音。

在任务 1 中，还对语音的每一帧进行了加窗的预处理，采用了汉宁窗，目的是让一帧信号的幅度在两端渐变到 0，以减轻语音帧与帧之间重叠部分对结果的影响，同时减轻频谱泄露的影响，保留语音信号的频率特性。

1.2. 算法描述

首先定义短时能量计算函数:

```
def compute_energy(signal):
    energy = 0
    for i in range(len(signal)):
        energy += signal[i]*signal[i]
    return energy
```

程序所使用的伪代码如下:

```
使用短时能量对一段音频f进行语音端点检测，得到标签序列
def process(f):
    params <- f's params 获取参数
    frames[i] <- f's i-th frame 分帧
    for i in f's all frames:
        frames[i] <- frames[i]*hanning 加窗
        energy[i] <- compute_energy(frames[i]) 计算能量
    result <- linear_classify(energy, threshoud) 线性分类器进行分类
    result <- remove_shorttime_silence(result) 去除短时静音
    return result

在测试集上得到结果
output <- result.txt 打开写入的结果文件
for f in test_set:
    result <- process(f)
    output <- result
```

除此之外，还尝试使用过零率以及短时能量和过零率的组合进行语音端点检测，过零率除过零率计算函数与短时能量计算函数不同外，整体结构与上面相似，定义过零率计算函数:

```
def compute_zero(signal):
    zcr = 0
```

```
    sign = np.sign(signal)
    for i in range(len(sign)-1):
        zcr += abs(sign[i]-sign[i+1])
    zcr /= 2
    return zcr
```

线性组合方法使用了 sklearn 库的逻辑回归线性分类器

```
label <- dev's label dev文件夹下所有的音频一起
data <- np.vstack((dev's zero_cross_rate, dev's
    energy)).T 组合两个特征
train,test <- train_test_split(data, label, size)
    划分数据集
data <- standardscaler(data) 标准化
model.fit(training data, training label) 模型训练
test model on test set 在测试集上评估
然后在test语音文件夹用此model进行预测
```

值得一提的是，在模型训练中，定义标签 0 和 1 的权重不同，对结果会造成较大的影响，可能是 0 和 1 在 dev 集上占比不同所产生的结果。在实际测试中，如果不在权重上加以约束，有很大机率出现将标签全部预测为 1 的情况，同时选 0 的权重为 0.7，1 的权重为 0.3 可以产生较好的结果。

1.3. 实验结果

首先针对所采用的短时能量预测算法，结果如表1。

表 1: 短时能量结果

名称	结果
准确率	0.92
auc	0.91
eer	0.08

除此之外，还测试了不同的能量分类阈值和去除短时静音的帧数阈值对结果的影响，结果如图1。

不同短时静音帧数对结果的影响如图2。

过零率预测算法结果如表2。

逻辑回归分类算法结果如表3。

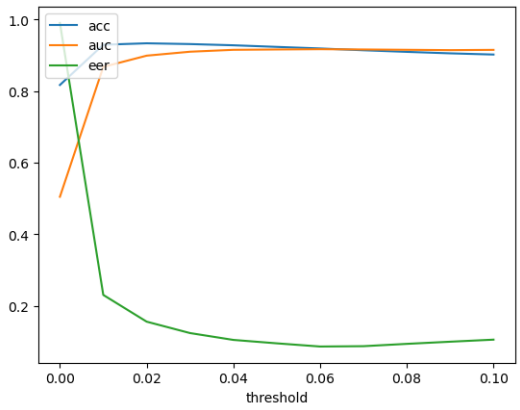

图 1: 结果随短时能量阈值变化图

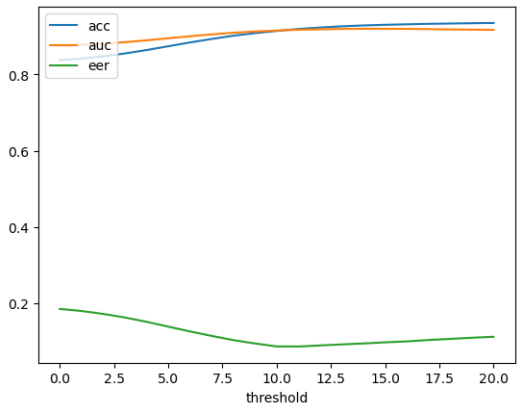

图 2: 结果随短时静音帧数阈值变化图

2. 基于统计模型分类器和语音频域特征的语音端点检测算法

2.1. 数据预处理及特征提取

任务 2 要求基于统计模型分类器和语音频域特征进行语音端点检测，本次任务在 MFCC, logFBank, 和二者组合中使用了 MFCC 特征进行语音端点检测。MFCC 特征 (梅尔倒谱系数) 是从低频到高频这一段频带内按临界带宽的大小由密到疏安排一组带通滤波器，对输入信号进行滤波，将每个带通滤波器输出的信号能量作为信号的基本特征，对此特征经过进一步处理后获得的语音输入特征。MFCC 是在 Mel 标度频率域提取出来的倒谱参数，Mel 标度描述了人耳频率的非线性特性，使其能更好的应对人耳对不同频率的声波有不同的听觉敏感度，具体表现为频率较低的声音容易掩蔽频率较高的声音，使得低音不易被人

表 2: 过零率结果

名称	结果
准确率	0.60
auc	0.58
eer	0.43

表 3: 逻辑回归分类结果

名称	结果
准确率	0.81
auc	0.82
eer	0.20

耳察觉。

logFBank 特征与 MFCC 特征相似，在提取流程上只是缺少了一个 DCT(离散余弦变换) 的过程，并直接在 FBank 上取对数作为结果。在提取到的语音特征上，logFBank 有着较高的特征相关性，但在判别度上稍逊于 MFCC 特征。

本任务使用了库 python_speech_features 中的 MFCC 特征提取函数，该函数会对输入的音频信号做预加重、分帧、加窗的预处理。其中分帧和加窗的预处理如任务 1 所述，预加重是对语音的高频部分进行加重, 去除口唇辐射的影响，用以补偿高频部分的衰减，增加高频语音部分的分辨率，它通过计算式3:

$$y(n) = x(n) - ax(n-1) \tag{3}$$

得以实现语音的预加重，其中 a 为系数，$x(n)$ 为第 n 个序列点。

2.2. 算法描述

本任务使用的算法将先对 train 数据集中的音频信号提取 MFCC 特征，在读取给定数据集的标签后训练 GMM 模型 (sklearn)，利用模型对 test 数据集进行端点检测。本次任务同样是在 MFCC、logFBank 和二者组合之中选取了 MFCC 特征进行端点检测。制作训练集的伪代码如下:

```
mfcc_train <- [] 初始化
mfcc_label <- []
for f in train_set:
    mfcc <- f's mfcc 提取mfcc
    label <- f's label(given) 提取给定label
    mfcc_train <- np.vstack((mfcc_train, mfcc)) 将f的特征放入总特征集中
    mfcc_label <- np.vstack((mfcc_label, label)) 将f的标签放入总标签集中
```

训练 GMM 模型以及在测试集中进行预测的伪代码如下:

```
gmm <- sklearn's gmm model 初始化模型
gmm.fit(train_set) 训练模型
for f in test_set:
    y_pred <- gmm.predict(f's mfcc) 对训练集进行预测
    y_txt <- prediction_to_vad_label(y_pred) 转化为文本格式
    result file <- y_txt 将结果写入文件
```

本任务还尝试使用了 DNN 模型 (pytorch)，但由于结果接近随机故未采用，代码在制作训练集上和 GMM 相似，在模型训练上的伪代码如下:

```
class DNN <- DNN
model <- DNN
criterion <- BCELoss
optimizer <- Adam 初始化
for epoch in range(epoch):
    output <- model(input) 训练集上预测
    loss <- criterion(output,label) 计算loss
    backward 反向传播
    optimizer 参数优化
```

在测试集上的预测伪代码与 GMM 相似。

2.3. 实验结果

首先，针对本任务所采用的 MFCC 特征与 GMM 模型，在开发集上得到的结果如如表4。

表 4: *GMM* 分类 *MFCC* 结果

名称	结果
准确率	0.78
auc	0.84
eer	0.26

使用 logFBank 特征与 GMM 模型得到的结果如表5。

使用 logFBank 特征与 MFCC 特征组合后利用 GMM 模型得到的结果如表6。

表 5: *GMM* 分类 *logFBank* 结果

名称	**结果**
准确率	0.77
auc	0.81
eer	0.25

表 6: *GMM* 分类特征组合结果

名称	**结果**
准确率	0.75
auc	0.82
eer	0.29