小组分工

本次 python 大作业，我们学习 python 中的许多第三方库：

1. 学习了 librosa 库中相关函数的内容，了解音频的处理方式，如本次实验的 mfcc 特征向量处理；
2. 学习了 numpy 和 matplotlib 库，了解到了如何去利用 python 对数据进行处理分析，并以图像的形式直观地呈现出来；
3. 学习了 sklearn 和 scipy 库，了解到了如何去利用 python 现有的库去完成机器学习的内容，如本次大作业中的 GMM 与 K-means 聚类分析。

我们不仅了解和学习了声音处理中的相关内容，而且掌握了如何利用概率来完成这些任务，通过这些学习到的知识，完成了程序的设计。

以下是本次实验的个人分工表：

实验分工			
	A	B	C
音频信号采集、分析	√	√	
实验背景分析	辅	辅	主
数据整理、分析		√	
文献查找与整理	√		√
绘制图像		√	
绘制流程图			√
实验题分析求解	√	√	√
模型建立	辅	主	辅
程序设计	辅	主	辅
代码调试	√	√	
论文撰写	主	辅	辅
论文排版	主		辅

基于 Python 实现的 GMM 算法的鼾声信号识别系统

摘要

打鼾是一种十分普遍的现象，它不仅困扰患者、影响同伴，还会对患者健康造成威胁。打鼾还是睡眠呼吸障碍最常见的症状，因此，通过识别不同鼾声的梅尔频率倒谱系数，区分不同人群的鼾声信号，以此为思路来开发一种以鼾声信号为基础的 OSAS 检测方法具有重要意义。

本文将基于 **Python 程序设计语言**，运用机器学习来训练鼾声信号模型。首先，我们采集了某同学预设睡眠时间段的音频信息作为训练集，并且在下载了国内外大量不同人群的鼾声信号数据，与此位同学的音频信息共同作为测试集。接下来，我们对收集的数据进行清洗、格式化，并提取其梅尔频率倒谱系数作为训练样本，根据训练样本，我们利用 **K-means 聚类算法**，发现训练样本的聚类散点图明显被分为为鼾声和非鼾声两类。然后，根据聚类结果将非鼾声剔除，得到鼾声训练集。

在训练集上，使用**混合高斯模型(GMM)**来训练此模型。并且，利用识别模型计算训练集中每个鼾声的生成概率，将生成概率按升序排列，由概率生成图和多次代码测试可得，概率门限为 0.65 时，效果最好。我们将测试集的**每一帧**的概率去和概率门限进行比对，实现判断这一帧是否属于鼾声帧并得到鼾声分布图。经模型检验，我们发现预测的精确度约为 **80%**，故认为预测效果较好。

通过本次实验，我们成功地开发了一种基于基于 Python 实现的 GMM 算法的鼾声信号识别系统，该系统能够有效地检测某段音频是否包含指定实验者鼾声信号，进一步研究可为临床诊断提供可靠依据。但是，此模型仍存在数据量有限的问题，且本实验仅基于几种机器学习的模型，可能会存在过拟合等问题，在未来的实验中，可以考虑使用其他机器学习模型，如支持向量机 (SVM) 或递归神经网络 (RNN)，并增加数据量，以进一步提高模型性能。

总之本次实验给我们带来了很多收获和启示，我们将继续努力，运用计算机科学与技术知识为临床医学的发展和人类健康的保障做出贡献。

关键词： Python 梅尔频率倒谱 K-means 聚类分析 混合高斯模型 鼾声识别

目录

一、 实验背景分析

1.1 实验背景

打鼾是一种十分普遍的现象，从目前的调查来看，大约有 20% ~ 40%的人群患有打鼾症状。[1]到了 60 岁，则有近 60%的男性和 40%的女性患有打鼾症状。[2]鼾声是入睡后发出的粗重鼻息声。睡眠时上气道咽腔肌肉张力相对降低，上气道塌陷。[3]当气流通过上气道的狭窄部位时，气流变得湍急并引起组织振动，从而出现鼾声。具体地，打鼾可以表征为软腭、咽壁、会厌和舌头的振动。[4][5]打鼾不仅困扰患者、影响同伴，还会对患者健康造成威胁。响亮的呼噜声可能会吵得旁人整夜不得安睡，使得同伴睡眠质量大大降低，甚至可能患上继发性睡眠障碍，造成工作生活的不和谐。

打鼾还是睡眠呼吸障碍最常见的症状。睡眠呼吸障碍，特别是阻塞性睡眠呼吸暂停综合征(OSAS)，阻塞性睡眠呼吸暂停综合症是一种伴有打鼾的呼吸疾病，它会造成白天嗜睡，头昏，头疼，记忆力衰退，乏力，反应迟钝，睡眠行为异常等症状。长期患有 OSAS 可能会引起高血压、冠心病、心衰、中风等多种疾病。

1.2 实验目的

本实验旨在开发一种基于机器学习的鼾声信号识别方法，用于某段音频是否包含实验者的鼾声片段，并输出对应鼾声片段所在时段。

1.3 研究意义

睡眠呼吸障碍是睡眠过程中出现的呼吸异常，包括睡眠呼吸暂停综合征、低通气综合征、慢性肺部及神经肌肉疾患引起的有关睡眠呼吸障碍等，其中以阻塞性睡眼呼吸暂停综合征在(OSAS)为主。有研究表明，OSAS 会造成白天嗜睡，头昏，头疼，记忆力衰退，乏力，反应迟钝，睡眠行为异常等症状。长期患有 OSAS 可引起高血压、冠心病、心衰、中风等多种疾病。医学界对此疾病的研究十分重视，且已取得了重大成果，目前诊断和评估打鼾的主要技术手段是导睡眠图，但是它需要患者整夜待在睡眠实验室中并连接大量的生理电极。由于具有非侵入式、廉价易用的特点，鼾声信号的声学分析方法已引起广泛关注和研究，并表现出极大的潜力。但是大多数仪器检测费用昂贵，不利于推广。因此，开发一种以鼾声信号为基础的 OSAS 检测方法具有重要意义。

二、 实验方法

以下为实验流程图：

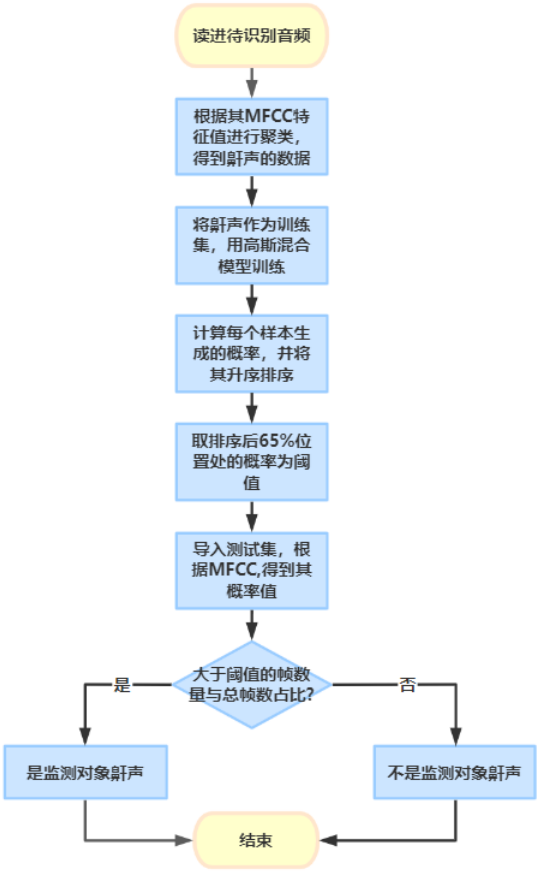

图 1 实验流程图

2.1 数据收集

在本次实验中，我们采集了某同学某三天在预设睡眠时间段的音频信息，并且在网站上下载了国内外大量不同人群的鼾声信号数据，以供进一步对此同学的鼾声信号进行分析。

2.2 数据预处理

我们对收集的数据进行清洗、格式化，并提取其梅尔频率倒谱系数作为训练样本；将训练样本利用 K-means 聚类算法分为鼾声和非鼾声两类，然后根据

聚类结果将非鼾声剔除，得到鼾声训练集。

2.3 模型训练

使用混合高斯模型（Gaussian mixture model, GMM），在数据预处理后得到的训练集上训练此模型。并且，利用识别模型计算训练集中每个鼾声的生成概率，将生成概率按升序排列，取其中靠前的预设百分比数据为该实验对象的鼾声生成概率门限；音频采集设备采集到有声段，然后提取该有声段的梅尔频率倒谱系数，再然后利用模型计算该有声段的生成概率，判断是否为实验对象的鼾声。

2.4 模型评估

将实验者的未参与训练的鼾声信号数据、网站上下载的国内外大量不同人群的鼾声信号数据作为测试集进行测试，并计算模型识别准确度，以此来分析模型准确性。

2.5 模型改进

根据实验结果，调整模型的结构和参数，并重复训练和评估，直到模型达到理想性能。

三、 实验模型假设

1) 假设此名同学的鼾声在采样期间未产生无明显变化；
2) 假设不同人群的鼾声信号均能被梅尔频率倒谱识别；
3) 假设采样音频无干扰性杂音；
4) 假设从网络上下载的鼾声采样真实可信。

四、 符号说明

符号	说明
$s(n)$	鼾声信号
$E(m)$	每个三角滤波器的输出能量
$C(n)$	梅尔频率倒谱系数特征
$p(x)$	GMM 的概率密度函数
π_k	第k个高斯模型的权重
$\gamma(i,k)$	计算第 i 个数据由第 k 个高斯分布函数产生的概率

五、 实验模型建立与求解

5.1 基于梅尔频率倒谱系数与 K-means 聚类算法的样本分类

5.1.1 梅尔频率倒谱(MFCC)的引入

在声音处理领域中，梅尔频率倒谱(MFCC)是基于声音频率的非线性梅尔刻度(mel scale)的对数能量频谱的线性变换,换言之，它是一个受不同人耳对具有不同频率的声波具有不同听觉灵敏度的事件启发的特征。MFCC 目前广泛应用于音频识别领域。

5.1.2 基于 MFCC 算法对音频信号的预处理

我们对采集到的音频信息进行分帧加窗处理，所述提取其梅尔频率倒谱系数具体包括如下步骤：

图 2 基于 MFCC 算法对音频信号的预处理流程图

具体步骤如下：

1) 对输入的信号进行预加重，分帧，并对一帧鼾声信号加窗函数
2) 进行快速傅里叶变换将其时域信息转换到频域

$$S(n)=FFT(s(n)\times w(n)) \tag{1}$$

3) 将经过傅里叶变换后的鼾声信号通过等Mel尺度的三角滤波器组得到每个三角滤波器的输出能量，其对数形式表示为

$$E(m)=ln\left(\sum_{n=0}^{N-1}\left(|S(n)|^2\times H_m(n)\right)\right),0\le m\le M \tag{2}$$

其中，M表示等Mel尺度的三角滤波器组中的等Mel尺度的三角滤波器数量，m是一个普通变量，其取值为0～M之间的整数；

4) 对$E(m)$进行离散余弦变换即可得到梅尔频率倒谱系数特征

$$C(n)=\sum_{n=0}^{N-1}E(m)cos\frac{\pi n(m-0.5)}{M},n=1,2,3,4,5,\dots,L \tag{3}$$

5) 得到特征向量

由下图可知，原本特征并不明显的音频波形图通过 MFCC 算法，对音频信号的预处理之后，通过一系列转化，得到了特征较为明显的 MFCC 特征向量。通过对特征向量进行 K-means 聚类分析，可以得到更精确的鼾声训练集

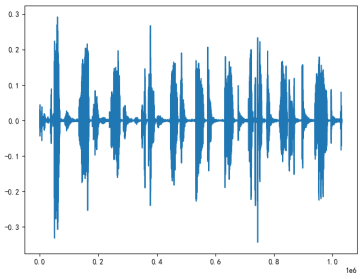

图 3 原始音频波形图

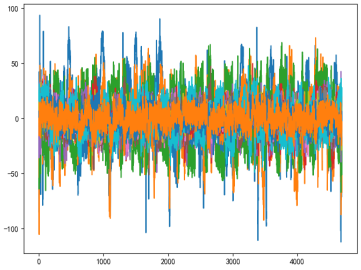

图 4 MFCC 特征向量图

批注 [h1]：这个特征向量好像算出来就够了，本质上 MFCC 就是一个 13 维的特征向量，然后通过这个特征向量来实现后面的聚类和机器学习。

批注 [h2R1]：贴一下我看到对 mfcc 比较好理解的是什么——本质上就是 13 维（加上 delta 和 accleration 共 69 维）的特征向量。
有什么用——特征向量本质上是一种编码，让机器“明白”（实际上是记住）频谱的信息

5.2 基于 K-means 聚类模型的信号划分

在对声音信号进行分类时，必须考虑到鼾声信号与非鼾声信号的相似程度。聚类算法作一种无监督的学习算法,便可以根据无标签样本点的对应数据及数据间的相似程度，将数据对象分组。一个良好的聚类效果，能使组内的样本点相似性大，且组间相似性小。因此，本题考虑采用 K-means 聚类算法[6]将声音信号分为鼾声与非鼾声两类。

5.2.1 K-means 聚类模型的原理

聚类算法是对一个表示n维向量的数据点集$D=\{x_i|i=1,\cdots,n\}$进行聚类划分，最终将集合D划分成k个类簇。聚类的依据，便是组内样本相似且组件差距大。聚类算法通常使用欧氏距离度量组间的相似性，其计算公式为

批注 [h3]：然后我们的模型就是根据 mfcc 得到的特征向量来做的吧

$$d(x,y)=\sqrt{\sum_{i=1}^{n}\left(x_i-y_i\right)^2} \tag{4}$$

此外，聚类算法以误差平方和作为聚类质量的目标函数。目标函数的计算公式为

$$SSE=\sum_{i=1}^{k}\sum_{n\subset c_j}^{n}\left|d\left(x,c_i\right)\right|^2 \tag{5}$$

通过不断优化目标函数，使原始数据点集被分为k个类别。而 K-means 算法是一种基于划分的、迭代性的的聚类算法。它是一种动态性的聚类，即对聚类过程不断重复迭代。其基本思想是:选择k个数据作最初的聚类中心（k为指定的类簇个数），计算各个数据点到每个初始聚类中心的欧式距离，计算方法为

$$d\left(x,c_i\right)=\frac{1}{2} \tag{6}$$

之后将数据点分配到其最近的聚类中心所在的集合，从而中形成一个簇。根据新形成的簇，可以更新其中心。重复此步骤，直到簇不发生变化或目标函数满足条件，聚类过程结束。

5.2.2 基于 K-means 聚类算法的信号分类模型

根据 K-means 算法的基本原理，将 mfcc 得到的特征向量作为数据点集进行数据划分，便可以得到聚类模型。聚类结果以及相应的聚类散点图如下：

批注 [h4]: 贴散点图就可以了吧，数据被很分为了两类巴拉巴拉的

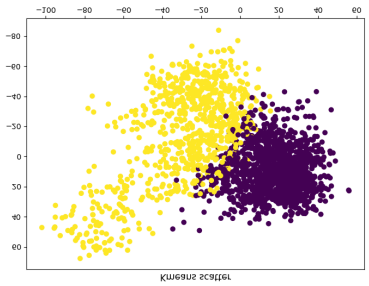

图 5 鼾声信号聚类散点图

根据上图，鼾声信号被明显分为两类，一类为鼾声信号，另一类为非鼾声信号。

批注 [h5]: 鼾声为一类，非鼾声为一类，如何区别？来一个水水的方法，和第一个做对比，假设第一个不是。或者 还有个方法，给每段音频开始的时候都加一个白噪音

5.3 混合高斯模型

5.3.1 混合高斯模型（Gaussian mixture model，GMM）的原理

批注 [h6]：这个原理我也不太懂，根据 psl 给的瞎扯就可以了吧，毕竟代码也只是抄了个函数 qwq，把图贴贴字抄抄改改应该问题不大

混合高斯模型的本质就是融合几个单高斯模型，来使得模型更加复杂，从而产生更复杂的样本。理论上，如果某个混合高斯模型融合的高斯模型个数足够多，它们之间的权重设定得足够合理，这个混合模型可以拟合任意分布的样本。

5.3.2 基于 GMM 算法的鼾声信号训练模型

根据以上实验结果，初始化 GMM。初始化 GMM 流程图如下左图，训练流程图如下右图。

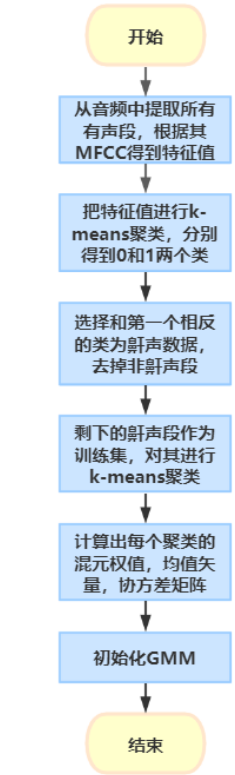

图 6 初始化 GMM 流程图

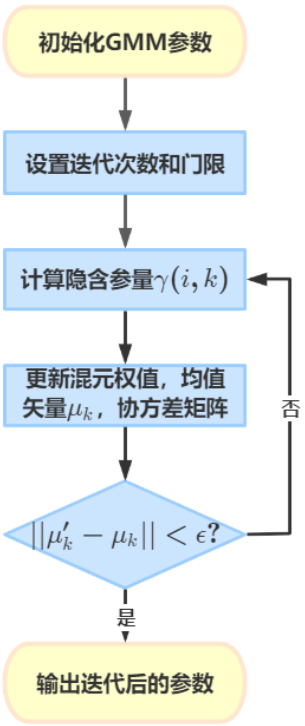

图 7 训练模型流程图

假设混合高斯模型由 K 个高斯模型组成（即数据包含 K 个类），则 GMM 的概率密度函数如下：

$$p(x)=\sum_{k=1}^{K}p(k)p(x|k)=\sum_{k=1}^{K}\pi_k N(x|u_{k,}\sum k) \tag{7}$$

其中，$p(x|k)=N(x|u_k,\sum k)$是第k个高斯模型的概率密度，可以看成，选定第k个高斯模型后，该模型产生x的概率；$p(x)=\pi_k$是第k个高斯模型的权重，称作选择第 k 个模型的先验概率，且满足$\sum_{k=1}^{K}\pi_k=1$。

$$N(x|u_k,\sum k)=\frac{exp\,(-\frac{1}{2}(x-u_k)\sum k\sum_k^{-1}(x-u_k)^T}{(2\pi)^{\frac{D}{2}}(|\sum k|)^{\frac{1}{2}}} \tag{8}$$

其中，D为数据的维数。

鼾声信号的特征矢量序列x_1，x_2，x_3，…x_n服从高斯混合模型分布，那么该序列的联合分布概率为：

$$P(x)=\prod_{i=1}^{n}p(x_i) \tag{9}$$

对$P(x)$取对数得：

$$L(x)=logP(x)=\sum_{t=1}^{n}log\big(p(x_i)\big)=\sum_{t=1}^{n}log\sum_{k=1}^{K}\pi_k N(x_i|u_{k,}\sum k) \tag{10}$$

但是，若直接对$L(x)$求导，计算极值非常麻烦，且这不利于计算机运算，因此，将其转化为对$L(x)$的下界函数求导：

根据詹森不等式$log\,E(x)\geq E(logx)$，可得：

$$\sum_{t=1}^{n}log\sum_{k=1}^{K}\pi_k N(x|u_{k,}\sum k)\geq\sum_{t=1}^{n}\sum_{k=1}^{K}\pi_k log\,(N(x_i|u_{k,}\sum k)) \tag{11}$$

上式右边分别对π_k、μ_k、Σ_k求偏导并令导函数为零可得参数重估函数：

$$u_k=\frac{1}{N}\sum_{t=1}^{n}\gamma(i,k)x_i \tag{12}$$

$$\sum k=\frac{1}{N_k}\sum_{t=1}^{n}\gamma(i,k)(x_i-u_k)(\mathrm{x_i}-\mathrm{u_k})^{\mathrm{T}} \tag{13}$$

$$\pi_{\mathrm{k}}=\frac{\mathrm{N_k}}{\mathrm{n}} \tag{14}$$

其中

$$\gamma(i,k)=\frac{\pi_k N(x_i|u_k,\sum k)}{\sum_{k=1}^{K}\pi_k N(x_i|u_k,\sum k)}$$

$$N_k=\sum_{i=1}^{n}\gamma(i,k)$$

其中，$\gamma(i,k)$表示第i个数据由第k个高斯分布函数产生的概率，N_k表示训练数据集中有N_k个数据由第k个高斯分布函数产生，n表示训练数据集的总个数。

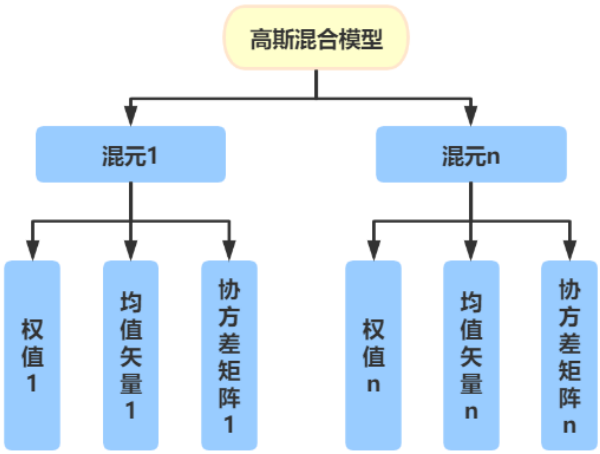

图 8 高斯混合模型混元分析

进一步地，通过如下步骤对高斯混合模型进行训练：

1) 计算第i个数据由第k个高斯分布函数产生的概率

$$\gamma(i,k)=\frac{\pi_k N(x_i|u_k,\sum k)}{\sum_{k=1}^{K}\pi_k N(x_i|u_k,\sum k)}$$

2) 计算GMM参数的估计值

$$u_k=\frac{1}{N}\sum_{i=1}^{n}\gamma(i,k)x_i$$

$$\sum k=\frac{1}{N_k}\sum_{i=1}^{n}\gamma(i,k)(x_i-u_k)(x_i-u_k)^T$$

$$\pi_k=\frac{N_k}{n}$$

3) 计算L(x)的值，若不收敛返回步骤1，收敛则退出。

收敛后，程序返回识别模型计算训练集中每个鼾声的生成概率值。并且，利用识别模型计算训练集中每个鼾声的生成概率，将生成概率按升序排列，取其中靠前的预设百分比数据为该实验对象的鼾声生成概率门限。

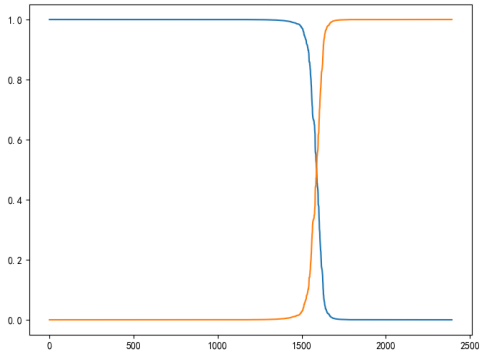

图 9 概率生成图

由概率生成图和多次代码测试可得，概率门限为 0.65 时，效果最好，故选用 0.65 作为概率门限值，然后我们将测试集的**每一帧**的概率去和概率门限进行比对，实现判断这一帧是否属于鼾声帧，得到如下鼾声分布图。

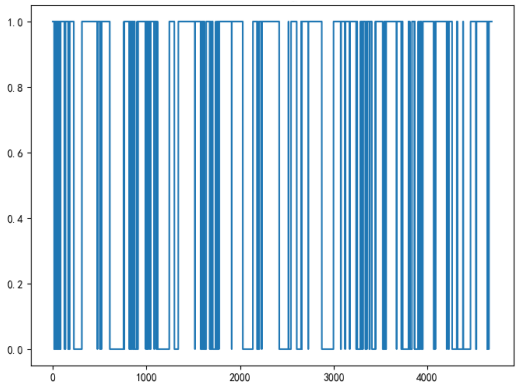

图 10 鼾声分布图

5.3.3 模型检验

将实验者的未参与训练的鼾声信号数据、网站上下载的国内外大量不同人群的鼾声信号数据作为测试集进行测试，并计算模型识别准确度，经过对比与测试，可发现实验精确度约为 80%，故认为预测效果较好。

```
  warnings.warn(
prob: [[1.00000000e+00 9.91978193e-13]
 [9.99999098e-01 9.02154180e-07]
 [1.00000000e+00 3.90727165e-10]
 ...
 [8.43883520e-01 1.56116480e-01]
 [9.92300390e-01 7.69960970e-03]
 [9.96413250e-01 3.58674954e-03]]
threshold: [0.20081391 0.79918609]
正确率: 0.796588486140725
```

图 11 模型检验过程

六、 实验结论

6.1 实验模型评价

1) 模型的优点：

通过本次实验，我们成功地开发了一种基于机器学习的鼾声信号识别方法，该方法能够有效地检测某段音频是否包含指定实验者鼾声信号，进一步研究，可为临床诊断提供可靠依据。本实验提出一种新的自适应的无监督的鼾声信号检测算法。检测算法主要分成两个层次三个步骤。第一个层次是有声段无声段的端点检测，识别出所有声音事件，这也是算法的第一步。

第二个层次是将识别出的声音事件区分为鼾声非鼾声信号，分为声音事件的特征提取和利用分类模型进行事件判别两步，这也是算法的第二步和第三步，主要是提取 MFCCs 特征并进行 K-means 聚类。

仿真结果表明这种新的检测算法具有优良的性能，准确度约为 80%。相比于监督学习算法，其无监督的特性无需大量的训练数据进行模型训练。

2) 模型的缺点：

首先，数据量有限，因此实验结果并不能完全代表所有情况。同时，在鼾声信号的检测中，特征和分类器的选择通常对检测性能有着比较大的影响。深度学习可以更好地表示数据的特征，并将特征和分类器融合在一个框架中，已在语音识别中取得了比较好的效果。可以从深度学习的角度来进行鼾声信号的检测。其次，在鼾声信号的分析中，发现患者整夜鼾声呈现明显的分层现象。但是由于数据标记的缺乏，仅仅给出导致这些变化的相关猜想。后续工作可以结合更充分的标记信息(如睡眠阶段、姿势等)，研究导致分层现象的生理原因

最后，本实验仅基于几种机器学习的模型，因此，模型的性能可能还有提升空间。目前还没有打鼾的力学建模，这方面的工作可能为鼾声分析带来新的角度。由于打鼾具有非线性的特点，可以从非线性动力学角度进行鼾声的相关分析。

6.2 实验模型优化

未来的研究可以尝试使用其他机器学习模型，如支持向量机 (SVM) 或递归神经网络 (RNN)，并增加数据量，以进一步提高模型性能。此外，可以尝试将该方法应用于其他类似的疾病的检测，如慢性肺部及神经肌肉疾患引起的睡眠呼吸障碍。在未来的研究中，我们建议进一步拓展数据量，并尝试使用不同的机器学习模型，如支持向量机 (SVM) 或递归神经网络 (RNN)，以进一步提高模型的性能和适用性。此外，建议在模型的结构和参数方面进行更多尝试，以找到更加优秀的模型，并不断优化模型的训练效率和应用效果。

6.3 实验反思

在实验过程中，我们经历了许多挑战和困难。首先，收集鼾声信号数据是一项繁琐而复杂的工作，因为鼾声信号来源于人体，并且具有很大的个体差异，因此需要对数据进行精心挑选和清洗。其次，我们在选择模型结构和调整参数方面也面临着许多挑战。需要不断尝试不同的模型结构和参数组合，并对模型在训练集和测试集上的表现进行评估，才能找到一个能够很好地拟合数据的模型。此外，我们还需要考虑模型的训练效率和推广应用的可行性。由于鼾声信号数据集较大，因此模型的训练过程需要耗费大量时间和计算资源。同时，模型的复杂度也会影响其在实际应用中的使用效率。因此，我们需要在提高模型性能的同时，尽可能减少模型的复杂度，使其能够被成功地应用于临床诊断。

总之，本次实验给我们带来了很多收获和启示，我们将继续努力，为开发基于鼾声信号的 OSAS 检测方法作出贡献，为临床医学的发展和人类健康的保障做出贡献。

参考文献

[1] Hoffstein V. Apnea and snoring: state of the art and future directions[J]. Acta Otorhinolaryngol. Belg., 2001, 56(2): 205-236.

[2] Lee B W V, Hill P D, Osbome J, et al. A simple audio data logger for objective assessment of snoring in the home[J]. Physiological measurement, 1999, 20(2): 119.

[3] Skatrud J B, Dempsey J A. Airway resistance and respiratory muscle function in snorers during NREM sleep[J]. Journal of Applied Physiology, 1985, 59(2): 328-335.

[4] Hoffstein V. Snoring[J]. CHEST Joumnal, 1996, 109(1): 201-222.

[5] Skatvedt O. Localization of site of obstruction in snorers and patients with obstructive sleep apnea syndrome: a comparison of fiberoptic nasopharyngoscopy and pressure mcasurements[J]. Acta oto-laryngologica, 1993, 113(1-2): 206-209.

[6] 王森,刘琛,邢帅杰.K-means 聚类算法研究综述[J/OL].华东交通大学学报:1-8[2022-09-18].

实验代码

```
import librosa #音频处理
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
from sklearn.mixture import GaussianMixture
from scipy.fftpack import dct
from matplotlib import cm
import os
def mfcc(data,sr):
    #预加重
    l = np.append(data[0],data[1:]-0.97*data[:-1])
    #分帧
    frame_size = 0.025 #30ms 为一帧
    frame_stride = 0.010 # 10ms 滑动（15ms 的交叠部分）
    frame_length  = int(round(frame_size * sr)) #帧的大小
    frame_step = int(round(frame_stride * sr))  #帧的滑动
    num_frames = int(np.ceil(float(np.abs(len(data) - frame_length))
/ frame_step)) #有多少帧
    indices = np.tile(np.arange(0, frame_length), (num_frames, 1)) +
np.tile(
    np.arange(0, num_frames * frame_step, frame_step), (frame_length,
 1)).T
    #indices 中的每一个元素为一帧,一帧有 550 个值，滑动 220
    frames = l[np.mat(indices).astype(np.int32, copy=False)]
    #为 frames 添加窗函数
    frames *= np.hamming(frame_length)
    #傅里叶变换
    NFFT = 512
    mag_frames = np.absolute(np.fft.rfft(frames, NFFT))
    #输出热力谱
    pow_frames = ((1.0 / NFFT) * ((mag_frames) ** 2))
    #转换 mel 频谱
    h = (2595 * np.log10(1 + (sr / 2) / 700)) #把最高频率转化为 mel 频率
    M = 40
    mel_points = np.linspace(0, h, M+2)  # 以 mel 频率为准划分坐标
    hz_points = (700 * (10 ** (mel_points / 2595) - 1))  # 再把坐标的点
转换为正常频率
    bin = np.floor((NFFT + 1) * hz_points / sr) #hz_point 的索引
    fbank = np.zeros((M, int(np.floor(NFFT / 2 + 1)))) #一个 40*257 的
容器
    #加三角窗
```

```
    for m in range(1, M + 1):
        f_m_minus = int(bin[m - 1])  # left
        f_m = int(bin[m])  # center
        f_m_plus = int(bin[m + 1])  # right
        for k in range(f_m_minus, f_m):
                fbank[m - 1, k] = (k - bin[m - 1]) / (bin[m] - bin[m
- 1])
            for k in range(f_m, f_m_plus):
                fbank[m - 1, k] = (bin[m + 1] - k) / (bin[m + 1] - bi
n[m])
    #对热力谱施加 mel 变换和三角窗
    filter_banks = np.dot(pow_frames, fbank.T)
    #数值稳定性
    filter_banks = np.where(filter_banks == 0, np.finfo(float).eps, f
ilter_banks)
    #以分贝为单位
    filter_banks = 20 * np.log10(filter_banks)
    #DCT 变换
    mfcc = dct(filter_banks, type=2, axis=1, norm='ortho')[:, 1 : 13]
    #减去均值
    mfcc -= (np.mean(mfcc, axis=0) + 1e-8)
    # 画热力图
    plt.imshow(np.flipud(mfcc.T), cmap=plt.cm.jet, aspect=0.2, extent
=[0, mfcc.shape[1], 0, mfcc.shape[0]])
    plt.show()
    return mfcc

os.chdir("C:/Users/28314/Desktop/codescat/Python")
audio, sr = librosa.load('test2.wav')
plt.figure(figsize=(8, 6))
plt.plot(audio)
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.title("声音波形图")
plt.show()

mfccs = mfcc(audio, sr)
X = mfccs
print('mfcc:', X)
plt.figure(figsize=(8, 6))
plt.plot(X)
plt.rcParams['font.sans-serif']=['SimHei']
```

```
plt.rcParams['axes.unicode_minus'] = False
plt.title("MFCC 特征向量图")
plt.show()
kmeans = KMeans(n_clusters = 2) #创建 k-means 聚类
kmeans.fit(X)
labels = kmeans.labels_
train_data = X[labels == (1 - labels[0]) ] #将鼾声的信息作为训练集
#这里因为不知道鼾声段属于 0 还是 1
#所以我们的处理是将鼾声段作为第一个数据取反的特征值
#因为第一段大概率不是鼾声，大概率是空白的噪音
model = GaussianMixture(n_components=2) #用高斯混合模型训练
model.fit(train_data)
probs = model.predict_proba(train_data) #计算训练集中每个样本生成的概率
probs = probs[np.argsort(probs[:,(1 - labels[0])]),:]     #升序排序
threshold = probs[int(len(probs) * 0.64)]

plt.figure(figsize=(8, 6))
plt.plot(probs)
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.title("概率分布图")
plt.show()

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.title('Kmeans scatter')
plt.show()

data, fs = librosa.load('test2.wav')
mfccs = mfcc(data, fs)
prob = model.predict_proba(mfccs)
cnt = 0
print('prob:',prob)
print('threshold:',threshold)
Anscnt = []
for i in range(len(prob)):
   if prob[i][1 - labels[0]] > threshold[1 - labels[0]]:
      Anscnt.append(1)
   else:
      Anscnt.append(0)
for i in range(len(prob)):
    if Anscnt[i] == labels[i]:
```

```
        cnt += 1
print('正确率:',cnt / len(prob))
plt.figure(figsize=(8, 6))
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.title("鼾声分布图")
plt.plot(Anscnt)
plt.show()
```