赛区评阅编号（由赛区组委会填写）：

2020 年高教社杯全国大学生数学建模竞赛

承　诺　书

我们仔细阅读了《全国大学生数学建模竞赛章程》和《全国大学生数学建模竞赛参赛规则》（以下简称“竞赛章程和参赛规则”，可从 http://www.mcm.edu.cn 下载）。

我们完全清楚，在竞赛开始后参赛队员不能以任何方式，包括电话、电子邮件、“贴吧”、QQ 群、微信群等，与队外的任何人（包括指导教师）交流、讨论与赛题有关的问题；无论主动参与讨论还是被动接收讨论信息都是严重违反竞赛纪律的行为。

我们完全清楚，在竞赛中必须合法合规地使用文献资料和软件工具，不能有任何侵犯知识产权的行为。否则我们将失去评奖资格，并可能受到严肃处理。

我们以中国大学生名誉和诚信郑重承诺，严格遵守竞赛章程和参赛规则，以保证竞赛的公正、公平性。如有违反竞赛章程和参赛规则的行为，我们将受到严肃处理。

我们授权全国大学生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号（从 A/B/C/D/E 中选择一项填写）：B

我们的报名参赛队号（12 位数字全国统一编号）：

参赛学校（完整的学校全称，不含院系名）：182

参赛队员 (打印并签名) ：1. 徐文杰

2. 张悦

3. 沈世杰

指导教师或指导教师组负责人 (打印并签名)：

（指导教师签名意味着对参赛队的行为和论文的真实性负责）

日期：　　年　　月　　日

（请勿改动此页内容和格式。此承诺书打印签名后作为纸质论文的封面，注意电子版论文中不得出现此页。以上内容请仔细核对，如填写错误，论文可能被取消评奖资格。）

赛区评阅编号：
(由赛区填写）

全国评阅编号：
（全国组委会填写）

2020 年高教社杯全国大学生数学建模竞赛

编 号 专 用 页

赛区评阅记录（可供赛区评阅时使用）：

评阅人						
备注						

送全国评阅统一编号:
（赛区组委会填写）

（请勿改动此页内容和格式。此编号专用页仅供赛区和全国评阅使用，参赛队打印后装订到纸质论文的第二页上。注意电子版论文中不得出现此页

2021 年高教社杯全国大学生数学建模竞赛

论文题目：基于 BP 神经网络的音乐流派鉴赏模型

摘　　要：

音乐分类是一项能够帮助用户找到自己喜欢类型歌曲的重要技术，本文基于 MFCC、过零率、频谱质心、谱滚降等 8 个指标建立了 BP 神经网络音乐流派鉴赏模型，对 2017 中国新歌声第二季前五期部分学员的歌曲进行音乐流派上的分类。

针对问题一，首先，本文通过预加重、分帧加窗技术对原始音频进行预处理，获得了一帧一帧的短时平稳信号；其次，基于声学和信号学，本文选取了 **MFCC**、**过零率**、**频谱质心**、**谱滚降**等 8 个指标建立了一种音乐特征指标体系，该指标体系可分为两类：声音的物理特性指标和人类的听觉感知指标；然后，提取音乐各类特征并组合，构造特征向量，提出基于 **BP 神经网络的音乐流派鉴赏模型**，对音乐分类效果进行分析。

针对问题二，通过利用问题一中提出的 BP 神经网络音乐流派鉴赏模型来实现对中国新歌声第二季前五期部分学员的歌曲有效的分类，最终在五种音乐流派中，流行乐的识别率最高达到 **90%**，同时古典乐的识别率也高达 **88%**。

关键词：音乐流派、音频特征、BP 神经网络

1. 问题重述

1.1 背景

随着计算机技术的不断成熟，信号处理的能力得到提升，其在音乐生成领域得到了成功应用，产生了大量的音乐。由于大多数人都喜欢听音乐，但是音乐的种类繁多，每一个人喜欢的音乐类型不同，如果事先对音乐信号类型进行分类和辨识，听众可以从音乐信号标签中选择自己想听的音乐，这样能够大幅度提升音乐的管理水平，因此音乐信号的分类和辨识成为人工智能领域的一个重要研究方向。

1.2 问题

问题一，对数字音乐，例如 WAV、MP3 等格式的音乐，使用音乐的部分（或全部）要素建立数学模型，给出音乐的分类方法。

问题二，利用问题1的分类方法，对2017中国新歌声第二季前五期部分学员的歌曲进行鉴赏评价，相关单曲可以由http://www.kugou.com/yy/html/search.html#searchType=song&searchKeyWord=中国新歌声第二季下载。

问题三，若时间允许的话，请在校园IP下，参照下列期刊或论文集的某篇（些）文章，利用数学建模的方法完成一篇以“数学与音乐”为主题的偏数学性质的文章。

2. 问题分析

2.1 问题一的分析

本文选取GTZAN数据集进行建模分析。首先利用预加重技术，去除音乐信号中的噪声并加强高频信号的比例，使信号波形趋于平坦。其次，音乐信号具有时变不平稳的特征，对信号进行分帧加窗，获得短时且平稳的音乐信号，来发现音乐特征。

本文基于声学和信号学，构建音乐特征指标体系，包括MFCC、过零率、频谱质心、谱滚降、色度频率、每分钟节拍、短时能量强度和时间相关性这8种特征。利用这8种特征，构建基于BP神经网络的音乐流派鉴赏模型对音乐流派进行识别。问题一的流程如图1。

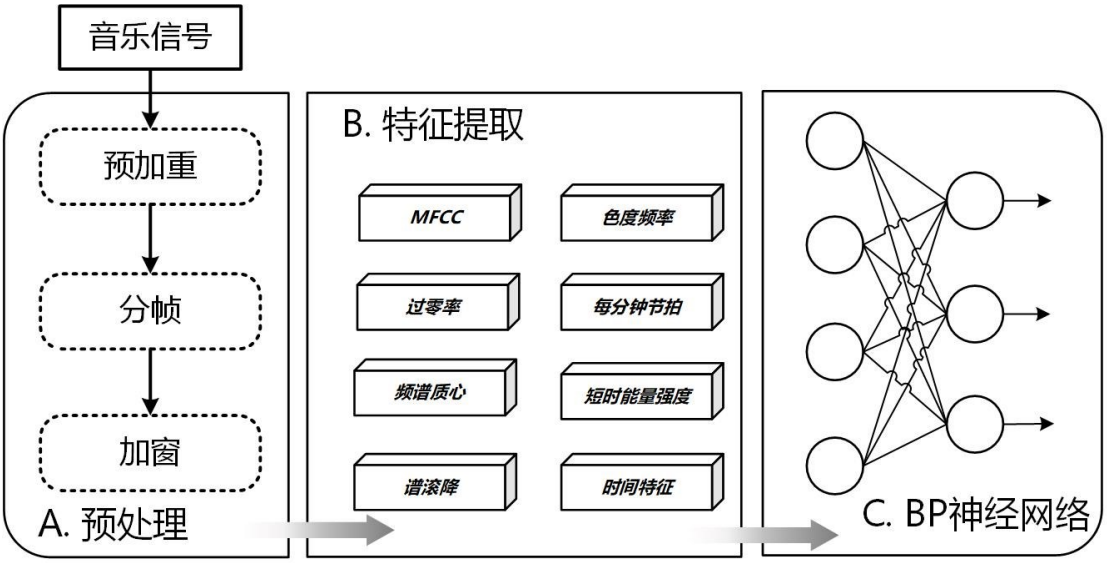

图 1. 问题一流程图

2.2 问题二的分析

基于问题一，利用音乐流派鉴赏模型对 2017 中国新歌声第二季前五期部分学员的歌曲进行音乐流派鉴赏。

3. 模型假设

3.1 假设音频在短时间内是变化缓慢的；

3.2 忽略主观因素对音乐风格分类的影响；

3.3 假设每个音乐分类都是明确的；

3.4 假设其他所发生的偶然因素对模型无影响

4. 符号说明

符号	意义
$x(n)$	待处理的数字化音乐信号
$y(n)$	经过滤波器处理之后得到的音乐信号
μ	预加重系数
$H(z)$	一阶高通数字滤波器
FC_t	频率质心
$w(m)$	窗函数

ω_{ij}	输入和隐含层的神经元权值
θ_j	隐含层的阈值
$f()$	传递函数
ω_{lj}	输出和隐含层的神经元权值
θ_l	输出层的阈值
k	学习次数
η	学习速率
TP	正类判定为正类
FP	负类判定为负类
FN	正类判定为负类
TN	负类判定为负类

5. 问题一的模型建立与求解

音乐信号混合了人声和乐器演奏声，而人声和采集语音信号的设备所带来的混叠、高频等因素对信号的质量产生影响。因此，本文对音乐信号进行预加重和分帧加窗，来获得更均匀、短时且平稳的信号。在此基础上，本文构建了8种音乐特征指标对信号进行特征提取，并构建BP神经网络对提取的8种特征进行建模训练，最终得到音乐流派鉴赏模型。

5.1 数据集

本文进行流派分类所使用数据集是音乐流派划分研究中较主流的GTZAN数据集[1]。GTZAN数据集包含蓝调、古典、乡村等10个音乐流派，每个流派包含100个30秒的音频，总计1000首。本文选取其中在中国更常见的蓝调、古典乐、嘻哈、流行乐与摇滚乐这五种流派的500首音乐进行建模分类。

5.2 预处理

一般预处理包括抗混叠滤波、预加重、数字化、加窗分帧等过程，而互联网上存储的音乐通常已被数字化处理，故对其通过预加重、归一化以及加窗分帧处理即可。

5.2.1 预加重

如图1，观察音乐信号原始时域波形图，由于人声和采集语音信号的设备所导致的混叠、高频等因素影响，从而导致了音乐信号的高低频比例不够均衡。因此，本文通过对信号进行预加重处理以此来均衡音乐信号，从而使得整个信号趋于平坦。

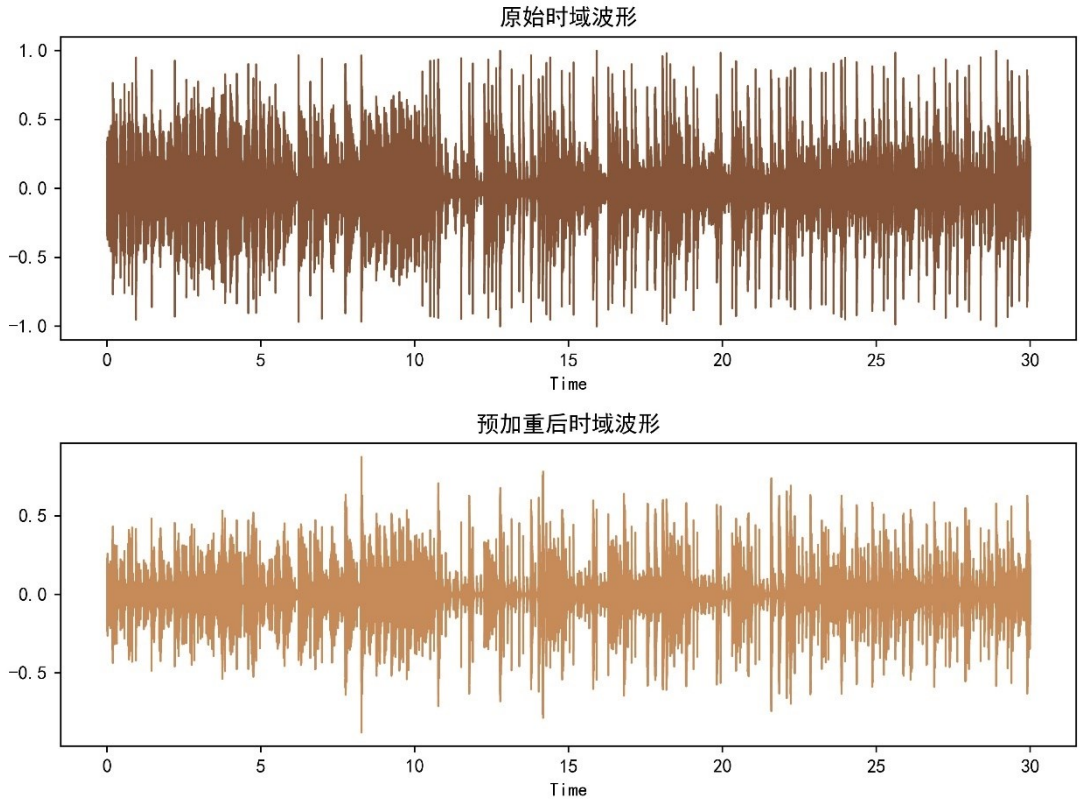

图2. 时域波形图

本文通过一阶的高通数字滤波器实现对音乐信号的预加重，该滤波器为：$H(z)=1-\mu z^{-1}$，其中参数μ为预加重系数，其取值范围为[0.9,1.0]，本文取$\mu=0.98$。一阶高通数字滤波器的预加重过程可以通过图3来表示，输出信号方程可以表示为：$y(n)=x(n)-\mu x(n-1)$。其中$x(n)$为输入的原始音乐信号，$y(n)$为通过滤波器后的输出信号。为了降低计算复杂度，还需要对$y(n)$实行数据归一化处理。

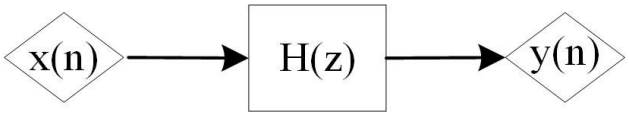

图3. 预加重滤波器示意图

5.2.2　分帧加窗

相关研究表明，音乐信号其本身存在着时变非平稳的特性，这种特性并不利于本文的研究[2]。虽然音乐信号具有时变特性，但对于10ms-30ms范围内的音乐片段来说，频谱波形基本维持平稳，能够被当成是一个短时且平稳的过程。因此，

本文截取音乐信号的每一帧并取帧长为30毫秒。而每帧的起始和末尾可能出现数据不连续的地方，导致帧数越多，与原始信号的误差越大。通过加窗使分帧后的每一帧音乐数据连续变化，防止区域内出现较大的峰起值。考虑到加窗处理会使得帧信号两端的部分削弱，相比于处于某一帧中央的信号，其重要性较弱。故本文在分针处理时将帧与帧之间相互重叠，相邻两帧之间起始位置的时间差称为帧移，本文取帧移为10毫秒。

本文选取汉宁窗作为窗口函数进行加窗处理，其所对应的数学表达式为：

$$w(n)=\begin{cases}0.5-0.5\cos(\dfrac{2\pi n}{N-1}), 0\le n\le N-1\\ 0, \text{其他}\end{cases} \tag{1}$$

通过以上所述的预处理，原先时变不平稳的音乐信号被分割成便于处理的以帧为单位的短时信号，且这些短时信号都可视为适合用于后期特征提取的平稳信号。这也为后续的特征提取做了铺垫。

5.3 音乐特征指标体系

音乐风格的识别主要通过其特征，因此特征提取直接决定了电子音乐分类的正确率的高低。本文建立了一种音乐特征指标体系，利用音乐信号处理的方式，提取音乐对应的音频特征。该指标体系可分为两类：（1） 声音的物理特性指标（如MFCC系数等）；（2）人类的听觉感知指标（如节拍，响度）。

5.3.1 声音的物理特性指标

a) 梅尔倒谱系数（MFCC）是一种关键的音乐信号辨识特征，人类的听觉特性可以被梅尔倒谱系数充分代表[3]。它先对音乐信号的每个短时分析窗进行傅里叶变换得到对应频谱并建立Mel倒谱的滤波器组，将此频谱通过滤波器组并进行倒谱分析，保留前20个值作为音乐信号的MFCC特征系数。图4展示了MFCC的提取过程。

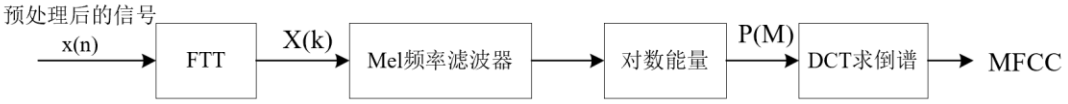

图4. MFCC 提取过程

b) 过零率指信号符号的变化比率，即每帧语音信号从正变为负或从负变为正的次数，在音乐检索和语音识别中被大量应用。通常对类似金属、摇滚等高冲击性的声音的具有更高的价值。过零率的计算公式如下：

$$\begin{aligned}Z_n&=\frac{1}{2}\sum_{m=-\infty}^{+\infty}\left|\text{sgn}\left[x(m)\right]-\text{sgn}\left[x(m-1)\right]\right|w(n-m)\\&=\frac{1}{2}\sum_{m=n}^{m+N-1}\left|\text{sgn}\left[x_N(m)\right]-\text{sgn}\left[x_N(m-1)\right]\right|\end{aligned} \tag{2}$$

其中，$x(m)$表示m帧的音乐信号，sgn[·]表示符号函数：

$$\mathrm{sgn}[x]=\begin{cases}1 & x\geq 0\\ -1 & x<0\end{cases} \tag{3}$$

c) 频谱质心指声音的“质心”所在位置，它按照声音的频率的加权平均值进行计算，它是度量音频亮度的指标，其计算方法如下：

$$FC_t=\frac{\sum_{n=1}^{N}M_t[n]\times n}{\sum_{n=1}^{N}M_t[n]} \tag{4}$$

d) 谱滚降通常表示的是谱能量在特定百分比时的频率，它是对数据信号形状的测量，本文取的百分比为85%，其公式如下：

$$R_r=\arg\left(\sum_{k=1}^{R_r}X_r(k)=0.95\cdot\sum_{k=1}^{N}X_r(k)\right) \tag{5}$$

其中k是频率下标:$k=1,2,...K;K$是频率下标的总个数；$X_r(k)$是第r帧信号在频率下标k处傅里叶变换的幅度。

e) 色度频率从音频的波形或功率谱图计算色谱图，将整个频谱投影到12个区间，代表着音乐八度音的12个不同的色度。

f) MFCC系数可以代表音乐信号在一帧片段内的短时特征。若是为了利用这个系数来和一首音乐在节奏、旋律等特征相联系，就应该于较长的时间段内分析。本文引入常用的均值法和方差法构建帧和帧之间的时间相关性特征，可以通过上述系数的均值和方差来表示，计算式如下:

均值：

$$\overline{m_n}=\frac{1}{N}\sum_{l=0}^{N-1}x_k(l) \tag{6}$$

方差：

$$\sigma^2=\frac{1}{N}\sum_{l=0}^{N-1}(x_k(l)-\overline{m_n})^2 \tag{7}$$

其中，$x_k(l)$ 为音乐信号第 l 帧时刻的值，N为抽取的总帧数。

5.3.2 人类的听觉感知指标

a) 每分钟节拍的概念就是在一分钟的时间段落之间，所发出的声音节拍的数量，一般可以用来表现一个音乐信号的节奏内容特征。本文采用Python中librosa库中librosa.beat.beat_track() 函数计算每分钟节拍数。

b) 音乐的时域特征有很多，本文选择短时能量特征，相对于普通声音，音乐的能量值相对更高，因此可以通过计算音乐信息中的每一帧的短时能量特征以表征音强这一声学特征的大小。

设音乐的信号$\{x(n)\}$的能量为E_n，其计算公式见式(8)：

$$E_n=\sum_{-\infty}^{+\infty}\left[x(m)\cdot w(n-m)\right]^2 \tag{8}$$

式中$w(m)$表示窗函数。

当音乐信号帧的长度为N时，E_n的计算公式变为：

$$E_n=\sum_{m=0}^{N-1}\left[x(m)\cdot w(n-m)\right]^2 \tag{9}$$

5.4 BP神经网络

基于音乐特征指标体系对音乐信号进行特征提取后，需要构建有效的分类器进行建模分析。目前，绝大部分的分类器均基于机器学习理论，如隐马尔可夫算法、神经网络、支持向量机等，它们均有各自的优点。相比于其他算法，BP神经网络的综合性能更优，具有优秀的泛化性能，在音乐分类研究中得到了广泛的应用[4]。综上，本文提出基于BP神经网络的音乐流派鉴赏模型，依据上述过程提取的音乐特征，建模训练并对音乐分类效果进行分析。

本文将提取出的 8 种音乐特征作为输入向量，即$X=(x_1,x_2,...,x_n)$，将 5 种音乐流派的标签通过 One-Hot 编码转换成 0 和 1 组成的五维向量作为输出层，即$Y=(y_1,y_2,y_3,y_4,y_5)$。输入层、输出层和一个隐含层一起组合成 BP 神经网络。

BP 神经网络的损失函数为交叉熵损失函数，其计算公式为：

$$L=\frac{1}{N}\sum_i L_i=-\frac{1}{N}\sum_i\sum_{c=1}^{M}y_{ic}\log(p_{ic}) \tag{10}$$

其中，M为类别的数量，y_{ic}为符号函数 0 或 1，p_{ic}为观测样本i属于类别c的概率。

5.4.1 模型训练及评价

5.4.1.1 评价指标

本文采用精确率、召回率、F_1值与混淆矩阵作为模型的精度评价指标，具体计算公式如下表 1 所示。混淆矩阵如下所示：

$$\begin{bmatrix} 18 & 1 & 0 & 0 & 1 \\ 3 & 15 & 0 & 2 & 0 \\ 1 & 1 & 15 & 0 & 3 \\ 0 & 0 & 1 & 18 & 1 \\ 5 & 0 & 3 & 0 & 12 \end{bmatrix}$$

表 1 模型评价指标

评价指标	公式
精确率	$Precision = \frac{TP}{TP + FP}$
召回率	$recall = \frac{TP}{(TP + FN)}$
F_1 值	$F_1 = 2\frac{Precision \times recall}{Precision + recall}$

5.4.1.2 训练及测试

本文将 5 种音乐流派进行处理后得到每一首音乐的输入特征，并将音乐流派都按 6:2:2 的比例划分，分别混合组成训练集，测试集，验证集。

基于上述过程，利用基于 BP 神经网络的音乐流派鉴赏模型来求解其对各流派音乐的识别率。如表 2 可知，在五种音乐流派中，流行乐的识别率最高，同时古典乐的识别率也高达 88%。这可能是由于流行乐与古典乐相较于其他音乐派别，在音乐的基本要素方面有着更加明显的差别。而蓝调的识别率最低，仅达 67%，这可能因为蓝调的节奏比较缓和轻松，容易被识别为古典乐。

表 2. 音乐流派分类结果识别率

音乐类别	精确率	召回率	F_1 值
蓝调	67%	90%	77%
古典乐	88%	75%	81%
嘻哈	79%	75%	77%
流行乐	90%	90%	90%
摇滚乐	71%	60%	65%

6. 问题二模型建立与求解

在本文中选取了蓝调、古典乐、嘻哈、流行乐与摇滚乐等五类不同的音乐构建了音乐流派鉴赏模型，对 2017 中国新歌声第二季前五期部分学员的 58 首歌曲进行鉴赏评价，部分结果见表 3。

表 3. 音乐流派分类结果

歌曲名称	分类
九妹-Ginga 金甲	嘻哈
于梓贝-模特	流行
⋮	⋮
张一腾-性别	流行
夏启明-Only You	古典

通过音乐流派鉴赏模型进行分析，58 首歌曲中，有 28 首为流行乐。这是因为，中国新歌声作为一场全民参加的素人音乐赛事，其中年轻人参赛众多。而年轻人熟悉并喜爱的歌曲大多为流行音乐。此外，嘻哈音乐开始兴起，也有 6 首嘻哈风格的音乐出现在参赛歌曲中。选手中青年人多偏多，大多数偏爱摇滚乐，因此摇滚乐的选择也不在少数。因为参赛选手年龄分布广，其余流派的音乐也各有部分选手演唱。

7. 模型评价与改进

7.1 模型优点

（1） 模型基于音乐特征指标体系进行特征选择，多角度的提取了音乐信号的不同特征。

（2） 基于 BP 神经网络的音乐流派鉴赏模型具有较好的分类精度和泛化性能。

7.2 模型缺点

蓝调、流行乐和古典音乐具有一定相似性，未选取合适的指标对三者进行区分，使蓝调的分类效果较差。

7.3 模型的改进

（1） 构建更多的音乐特征指标，以提高风格相似音乐流派的分类精度。

（2） 在 BP 神经网络引入优化算法，以提高模型的预测性能与精度。

8. 参考文献

[1] Tzanetakis G, Essl G, Cook P. Automatic Musical Genre Classification Of Audio Signals. 2001.

[2]唐霞，张晨曦，李江峰．基于深度学习的音乐情感识别[J]．电脑知识与技术，2019, v.15(11):238-243.

[3] Marius, Kaminskas, and, et al. Contextual music information retrieval and recommendation: State of the art and challenges[J]. Computer Science Review, 2012, 6(2-3):89-119.

[3] Dagan I, Engelson S P. Committee-Based Sampling For Training Probabilistic Classifiers[J]. Machine Learning Proceedings, 1995:150-157.

[4]刘建辉, 曾丽辉, 许金凤,等. 基于最小二乘支持向量机的乐器音乐分类[J]. 华东交通大学学报, 2009(06):66-70.

9. 附录

```
import os
import pickle
import random
import operator
import math
import numpy as np
import pandas as pd
import librosa.display
import matplotlib.pyplot as plt

# 设置字体为黑色
font_name = "SimHei"
# 显示中文标签
plt.rcParams['font.family'] = font_name
plt.rcParams['axes.unicode_minus'] = False

import librosa
signal, sam_rate = librosa.load('E:/实验数据/音频数据/genres/blues/blues.00000.wav')



plt.figure(figsize=(20, 10))
plt.subplot(3,1,1)
plt.title("时域可视化波形")
librosa.display.waveshow(signal, sr=sam_rate,color='#865439')

# 纵轴表示频率（从 0 到 10kHz），横轴表示剪辑的时间。
plt.subplot(3,1,2)
plt.title("频谱图")
X = librosa.stft(signal)
Xdb = librosa.amplitude_to_db(abs(X))
librosa.display.specshow(Xdb, sr=sam_rate, x_axis='time', y_axis='hz')
plt.colorbar()

plt.subplot(3,1,3)
plt.title("波形的幅度包络")
librosa.display.waveplot(signal, sr=sam_rate,color='#C68B59')

# 信号加重
def signal_empha(signal,u= 0.98):
    emphasized_signal = np.append(signal[0], signal[1:] - u *signal[:-1])
    return emphasized_signal
```

```
#分帧函数
def enframe(signal, nw, inc, windown):
    '''将音频信号转化为帧。
    signal:原始音频型号
    nw:每一帧的长度(这里指采样点的长度，即采样频率乘以时间间隔)——30/1000*22050
    inc:相邻帧的间隔
    '''
    signal_length=len(signal) #信号总长度
    if signal_length<=nw: #若信号长度小于一个帧的长度，则帧数定义为 1
        nf=1
    else: #否则，计算帧的总长度
        nf=int(np.ceil((1.0*signal_length-nw+inc)/inc))

    pad_length=int((nf-1)*inc+nw) #所有帧加起来总的铺平后的长度
    zeros=np.zeros((pad_length-signal_length,)) #不够的长度使用 0 填补，类似于 FFT 中的扩充数组操作
    pad_signal=np.concatenate((signal,zeros)) #填补后的信号记为 pad_signal
    indices=np.tile(np.arange(0,nw),(nf,1))+np.tile(np.arange(0,nf*inc,inc),(nw,1)).T    #相当于对所有帧的时间点进行抽取，得到 nf*nw 长度的矩阵
    indices=np.array(indices,dtype=np.int32) #将 indices 转化为矩阵
    frames=pad_signal[indices] #得到帧信号
    win=np.tile(windown,(nf,1))   #window 窗函数，这里默认取 1
    return frames*win     #返回帧信号矩阵

for i in label_list:
    file_inf = pd.read_csv(os.path.join(label_path,i),header=None).to_numpy()
    music_array = []
    for j in file_inf:
        signal, sr = librosa.load(j[0])
        #  加重
        emphasized_signal = signal_empha(signal,u=0.98)
        #  汉宁窗
        # winfunc = np.hanning(660)
        #  分帧加窗
        # divided_signal = enframe(emphasized_signal,660,220,winfunc)
        fea_arr = np.array([], dtype=np.float64)
        # MFCC
        mfccs = librosa.feature.mfcc(emphasized_signal, sr)
        #  过零率
        zero_cross = librosa.feature.zero_crossing_rate(emphasized_signal)
        #  频谱质心
        cent = librosa.feature.spectral_centroid(emphasized_signal, sr=sr)
        #  色度频率
```

```
        chroma = librosa.feature.chroma_stft(emphasized_signal, sr=sr)
        # 谱滚降
        rolloff = librosa.feature.spectral_rolloff(emphasized_signal, sr=sr, roll_percent=0.95)
        # 每分钟节拍
        tempo, beats = librosa.beat.beat_track(signal,sr)
        # 短时平均能量
        energy = np.square(emphasized_signal).sum()
        # 时间特征

        fea_arr = np.append(fea_arr,np.mean(zero_cross))
        fea_arr = np.append(fea_arr,np.mean(cent))
        fea_arr = np.append(fea_arr,np.mean(chroma))
        fea_arr = np.append(fea_arr,np.mean(rolloff))
        fea_arr = np.append(fea_arr,np.mean(mfccs))
        fea_arr = np.append(fea_arr,np.var(mfccs))
        fea_arr = np.append(fea_arr,np.mean(mfccs,axis=1).flatten())
        fea_arr = np.append(fea_arr,np.mean(energy))
        fea_arr = np.append(fea_arr,tempo)

        if j[1] == 'bl':
            fea_arr = np.append(fea_arr,'blues')
        if j[1] == 'cl':
            fea_arr = np.append(fea_arr,'classical')
        if j[1] == 'co':
            fea_arr = np.append(fea_arr,'country')
        if j[1] == 'di':
            fea_arr = np.append(fea_arr,'disco')
        if j[1] == 'hi':
            fea_arr = np.append(fea_arr,'hiphop')
        if j[1] == 'ja':
            fea_arr = np.append(fea_arr,'jazz')
        if j[1] == 'me':
            fea_arr = np.append(fea_arr,'metal')
        if j[1] == 'po':
            fea_arr = np.append(fea_arr,'pop')
        if j[1] == 're':
            fea_arr = np.append(fea_arr,'reggae')
        if j[1] == 'ro':
            fea_arr = np.append(fea_arr,'rock')
        fea_arr[:-1] = fea_arr[:-1].astype(np.float32)
        music_array.append(fea_arr)
    music_array = np.vstack(music_array)
    np.save(os.path.join(output_path,i[0:2]),music_array)
```

```
import librosa
import pandas as pd
import numpy as np
import os

aim_path = 'E:/实验数据/音频数据/genres/array_Data'
aim_list = os.listdir(aim_path)

music_data = []
music_data_test = []
for i in aim_list:
    path = os.path.join(aim_path,i)
    dt = np.load(path)
    music_data.append(dt[:-20,:])
    music_data_test.append(dt[-20:,:])
music_data = np.vstack(music_data)
music_data_test = np.vstack(music_data_test)

from sklearn import preprocessing
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

# 载入数据集
feature = music_data[:, :-1].astype(np.float32)
label = music_data[:, -1]
feature_test = music_data_test[:, :-1].astype(np.float32)
label_test = music_data_test[:, -1]
scaler_1 = preprocessing.StandardScaler()
feature = scaler_1.fit_transform(feature)
feature_test = scaler_1.transform(feature_test)
scaler_2 = preprocessing.LabelBinarizer()
label = scaler_2.fit_transform(label)
label_test = scaler_2.transform(label_test)
dataset_net = np.concatenate((feature, label), axis=1)
dataset_net_test = np.concatenate((feature_test, label_test), axis=1)
np.random.shuffle(dataset_net)
np.random.shuffle(dataset_net_test)

# 将数据拆分成训练集和测试集
X_train, y_train = dataset_net[:, :-5], dataset_net[:, -5:]
X_test,y_test = dataset_net_test[:, :-5], dataset_net_test[:, -5:]

#导入相关库
```

```
import tensorflow as tf
from tensorflow.keras import activations
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.callbacks import EarlyStopping

# BP 神经网络
model = Sequential()
model.add(Dense(32,activation='sigmoid'))
model.add(Dense(100,activation='sigmoid'))
model.add(Dense(5,activation='sigmoid'))
model.compile(loss=tf.keras.losses.CategoricalCrossentropy(),optimizer='adam')
early_stop = EarlyStopping(monitor='loss', patience=50, verbose=1)

history = model.fit(X_train,
                    y_train,
                    epochs=10000,
                    batch_size=128,
                    shuffle=True,
                    validation_split=0.2
                    verbose=2,
                    callbacks=[early_stop])

# plot train and validation loss
plt.figure(figsize=(8,8))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model train vs validation loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','validation'], loc='upper right')
plt.show()

# 训练完成之后，使用测试数据对模型进行测试
pred = model.predict(X_test)

pre_y = np.argmax(pred, axis=1)
test_y = np.argmax(y_test,axis=1)

# 查看模型预测结果与真实标签之间的报告
print(classification_report(test_y, pre_y))
# 查看模型预测结果与真实标签之间的混淆矩阵
print(confusion_matrix(test_y, pre_y))
```

```
# 验证
def proprecess(signal,sr):
    emphasized_signal = signal_empha(signal, u=0.98)
    fea_arr = np.array([], dtype=np.float64)
    # MFCC
    mfccs = librosa.feature.mfcc(emphasized_signal, sr)
    # 过零率
    zero_cross = librosa.feature.zero_crossing_rate(emphasized_signal)
    # 频谱质心
    cent = librosa.feature.spectral_centroid(emphasized_signal, sr=sr)
    # 色度频率
    chroma = librosa.feature.chroma_stft(emphasized_signal, sr=sr)
    # 谱滚降
    rolloff = librosa.feature.spectral_rolloff(
    emphasized_signal, sr=sr, roll_percent=0.95)
    # 每分钟节拍
    tempo, beats = librosa.beat.beat_track(signal, sr)
    # 短时平均能量
    energy = np.square(emphasized_signal).sum()
    # 时间特征
    fea_arr = np.append(fea_arr, np.mean(zero_cross))
    fea_arr = np.append(fea_arr, np.mean(cent))
    fea_arr = np.append(fea_arr, np.mean(chroma))
    fea_arr = np.append(fea_arr, np.mean(rolloff))
    fea_arr = np.append(fea_arr, np.mean(mfccs))
    fea_arr = np.append(fea_arr, np.var(mfccs))
    fea_arr = np.append(fea_arr, np.mean(mfccs, axis=1).flatten())
    fea_arr = np.append(fea_arr, np.mean(energy))
    fea_arr = np.append(fea_arr, tempo)
    return fea_arr

test_path = 'E:/实验数据/音频数据/测试'
label_list = os.listdir(test_path)
test_array = []
for i in label_list:
    song,sr = librosa.load(os.path.join(test_path,i))
    feature = proprecess(song,sr)
    test_array.append(feature)
test_array = np.vstack(test_array)
t = scaler.transform(test_array)
p1 = model.predict(t)
result= np.argmax(p1,axis=1)
dist = 0
for i in range(len(result)):
```

```
        if result[i] == 4:
            dist += 1
dist = []
for i in range(len(result)):
        if result[i] == 0:
            dist.append(np.append(label_list[i][:-4], '蓝调'))
        elif result[i] == 1:
            dist.append(np.append(label_list[i][:-4], '古典'))
        elif result[i] == 2:
            dist.append(np.append(label_list[i][:-4], '嘻哈'))
        elif result[i] == 3:
            dist.append(np.append(label_list[i][:-4], '流行'))
        elif result[i] == 4:
            dist.append(np.append(label_list[i][:-4], '摇滚'))
dist = np.vstack(dist)
pd.DataFrame(dist)
```