中国人民大学
RENMIN UNIVERSITY OF CHINA

基于 HMM 的语音数字识别

报告题目：基于 HMM 的语音数字识别

姓名：马正一

学号：2019101404

专业：信息学院 2019 级 计算机应用技术

联系电话：18810971511

电子信箱：zymaa@ruc.edu.cn

课程：言语信息处理

提交日期： 2020 年 01 月 03 日

1 任务描述

基于数字语音数据集，编写代码，使用 HMM 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=0.6，pyaudio，scikit-learn=0.18.1，hmmlearn=0.2.3，scipy。

3 实验方案

3.1 MFCC 特征提取

我们使用课程提供的英文数据集，包括数字 0-9 共 150 个 wav 格式的音频文件。我们使用 Python 的 wav 包读取 wav 文件，使用 python-speech-features 获得每条音频数据的 39 维 MFCC 特征。我们在本实验中对加入一阶导与二阶导的 39 维特征进行了实验。具体来说，MFCC 特征提取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。

在实际编写代码时，我们在 python 代码中编写了特征提取函数，返回每个数据样本的 39 维 MFCC 特征。

```
# 特征提取，feat = compute_mfcc(wadict[wavid])
def compute_mfcc(file):
    # print(file)
    fs, audio = wavfile.read(file)
    mfcc_feat = mfcc(audio, samplerate=fs,numcep=13, winlen=0.025,
winstep=0.01,nfilt=26, nfft=2048, lowfreq=0, highfreq=None, preemph=0.97)
    d_mfcc_feat = delta(mfcc_feat, 1)
    d_mfcc_feat2 = delta(mfcc_feat, 2)
    feature_mfcc = np.hstack((mfcc_feat, d_mfcc_feat))
    return feature_mfcc
```

3.2 HMM 分类

在获取了 MFCC 特征之后，我们将编写基于 HMM 的分类算法。我们使用了 hmmlearn 的 GMMHMM 基于高斯混合分布的隐马尔科夫模型来编写程序。在本实验中，我们面的是一个十分类问题。我们将训练十个单核 HMM。对于每个数字的所有样本，我们用这部分训练数据训练同一个单核 HMM。在测试阶段，我们将每一个待分类样本输入至每一个 HMM，得到在该 HMM 下的对应评分，即该 HMM 对应数字的评分。我们对十个评分进行降序排序，将评分最高的 HMM 对应数字作为预测标签进行输出。

Hmmlearn 中对于 GMMHMM 模型的定义如下

class hmmlearn.hmm.**GMMHMM**(*n_components=1*, *n_mix=1*, *min_covar=0.001*, *startprob_prior=1.0*, *transmat_prior=1.0*, *weights_prior=1.0*, *means_prior=0.0*, *means_weight=0.0*, *covars_prior=None*, *covars_weight=None*, *algorithm='viterbi'*, *covariance_type='diag'*, *random_state=None*, *n_iter=10*, *tol=0.01*, *verbose=False*, *params='stmcw'*, *init_params='stmcw'*)

- n_components: 混合高斯模型个数，默认为 1
- n_mix：HMM 中的状态个数，默认为 1
- covariance_type: 协方差类型，包括｛‘full’，‘tied’， ‘diag’，‘spherical’｝四种，full 指每个分量有各自不同的标准协方差矩阵，完全协方差矩阵（元素都不为零）， tied 指所有分量有相同的标准协方差矩阵（HMM 会用到），diag 指每个分量有各自不同对角协方差矩阵（非对角为零，对角不为零）， spherical 指每个分量有各自不同的简单协方差矩阵，球面协方差矩阵（非对角为零，对角完全相同，球面特性），默认‘full’ 完全协方差矩阵
- startprob_prior：Dirichlet 的先验分布参数
- algorithm：解码算法，在”viterbi”与”map”中选其一，默认为”viterbi”。
- random_state ： 随机数发生器 n_init: 初始化次数，用于产生最佳初始参数，默认为 1
- tol:EM 算法的收敛阈值
- verbose ： 使能迭代信息显示，默认为 0，可以为 1 或者大于 1（显示的信息不同）
- init_params: ｛‘kmeans’， ‘random’｝, defaults to ‘kmeans’. 初始化参数实现方式，默认用 kmeans 实现，也可以选择随机产生
- params:决定训练阶段哪个参数被更新

3.3 HMM 实现

具体在编写代码时，我们编写核心模型类 Model，用于维护 10 个 HMM 高斯混合分布-隐马尔科夫模型。对于每个数字的训练样本，我们调用每个 GMMHMM 的 fit 函数进行模型训练。在测试阶段，我们调用 GMMHMM 的 score 函数，使用十个 HMM 分别进行评分，并对评分进行降序排序取最高分对应 label，作为预测输出。

```
class Model():
    def __init__(self, CATEGORY=None, n_comp=1, n_mix = 1, cov_type='full',
n_iter=100000):
        super(Model, self).__init__()
        print(CATEGORY)
        self.CATEGORY = CATEGORY
        self.category = len(CATEGORY)
        self.n_comp = n_comp
        self.n_mix = n_mix
        self.cov_type = cov_type
        self.n_iter = n_iter
        # 关键步骤，初始化 models，返回特定参数的模型的列表
        self.models = []
        for k in range(self.category):
            # model = hmm.GMMHMM(n_components=self.n_comp, n_mix = self.n_mix,
            #                    covariance_type=self.cov_type)
            model = hmm.GMMHMM(n_components=self.n_comp, n_mix =
self.n_mix,covariance_type=self.cov_type )
            self.models.append(model)

    # 模型训练
    def train(self, wavdict=None, labeldict=None):
        for k in range(10):
            subdata = []
            model = self.models[k]
            for x in wavdict:
                if labeldict[x] == self.CATEGORY[k]:
                    print("k=",k,wavdict[x])

                    mfcc_feat = compute_mfcc(wavdict[x])
                    # print(mfcc_feat)
                    # print(mfcc_feat)
                    model.fit(mfcc_feat)

    # 使用特定的测试集合进行测试
```

```
    def test(self, filepath):
        result = []
        for k in range(self.category):
            subre = []
            label = []
            model = self.models[k]
            mfcc_feat = compute_mfcc(filepath)
            # 生成每个数据在当前模型下的得分情况
            re = model.score(mfcc_feat)
            subre.append(re)
            result.append(subre)
        # 选取得分最高的种类
        result = np.vstack(result).argmax(axis=0)
        # 返回种类的类别标签
        result = [self.CATEGORY[label] for label in result]
        return result


    def save(self, path="models.pkl"):
        # 利用 external joblib 保存生成的 hmm 模型
        joblib.dump(self.models, path)


    def load(self, path="models.pkl"):
        # 导入 hmm 模型
        self.models = joblib.load(path)
```

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

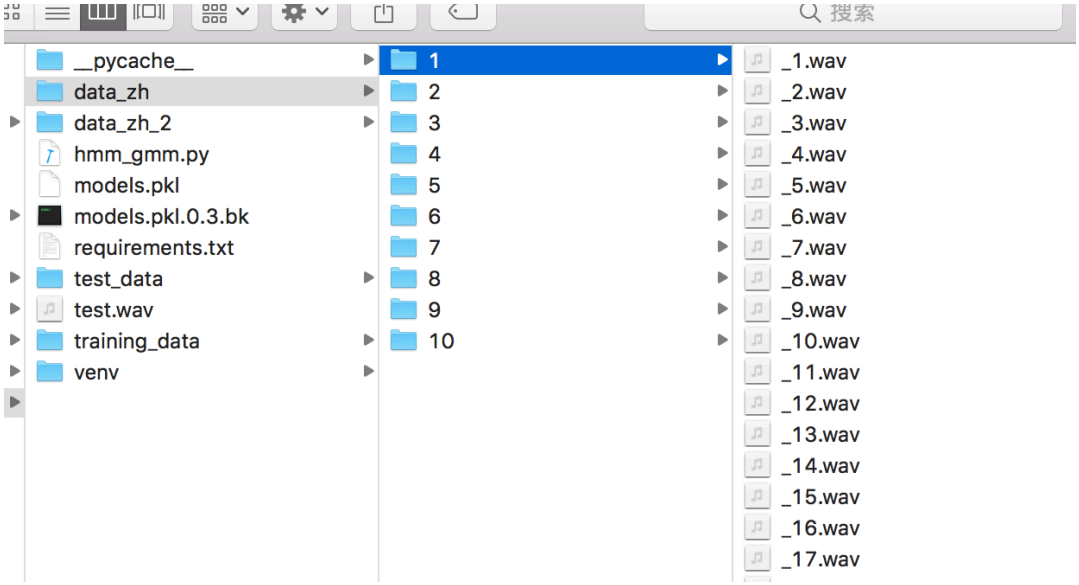
```
conda create -n HMM python=3.6 numpy pyaudio scipy hmmlearn scipy #也可以使用 pip
conda activate HMM
pip install -r requirements.txt
```

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/HMM_Digital_Voice_Recognition/ HMM_DVR
  cd HMM_DVR
  unzip /tmp/dataset.zip -d ./  # dataset.zip 是从百度网盘下载的数据
  python hmm_gmm.py
```

5 实验结果及运行截图

本实验在课程提供的中文数据集下，使用 285 条样本进行训练，30 条样本进行测试，对测试的 30 个样本可正确识别其中 19 个样本，准确率达到 40%。



原始数据文件

模型训练阶段

模型测试，准确率 12/30 = 40%