组　　号　第一组　　密级 公开

年　　份　2024　　成绩

福州大学

系统综合实践课程实验报告

实验五 基于 GMM+HMM 的语音识别实现

所在院系：计算机与大数据学院

专业年级：2021 级计算机科学与技术（实验班）

学　　号：102101141

姓　　名：高孙炜

任课教师：刘菀玲

联系方式：13599396788

邮件地址：1845613403@qq.com

1 实验内容
2 实验环境
3 实验过程 孤立词
3.1 MFCC 特征提取
3.2 GMM-HMM初始化
3.3 K-Means聚类
3.4 **生成 GMM 参数**
3.5 整体训练流程
3.6 测试
4 实验过程 连续词
4.1 导入所需库 配置路径
4.2 特征提取函数
4.4 创建GMMHMM模型
4.5 读取训练数据 训练模型
4.6 测试
5 实验总结

1 实验内容

主要内容为基于Python3.7和框架scipy、python_speech_features、hmmlearn以及numpy等工具进行简单的孤立词和连续词识。经过实验之后，让同学们掌握对模型GMM-HMM的理解以及应用GMM-HMM进行孤立词和连续词识别的实践操作。

2 实验环境

- MindSpore1.1.1+Python3.7
- scipy1.14.1
- numpy1.18.4
- python_speech_features0.6、hmmlearn

3 实验过程 孤立词

3.1 MFCC 特征提取

```
def extract_MFCC(wav_file):
        # 读取音频数据
        y, sr = librosa.load(wav_file, sr = 8000)
        # 提取特征
        fea = librosa.feature.mfcc(y=y, sr=sr, n_mfcc= 12, n_mels = 24, n_fft =
256, win_length = 256, hop_length = 80, lifter=12)
        # 进行正则化
        mean = np.mean(fea, axis = 1, keepdims = True)
        std = np.std(fea, axis = 1, keepdims = True)
        fea = (fea - mean) / std
        # 添加1阶差分
        fea_d = librosa.feature.delta(fea)
        fea = np.concatenate([fea.T, fea_d.T], axis = 1)
        return fea
```

3.2 GMM-HMM初始化

```
def init_parm_hmm(collect_fea, N_state, N_mix):
        pi = np.zeros(N_state)
        pi[0] = 1

        A = np.zeros([N_state, N_state])
        for i in range(N_state - 1):
                A[i, i] = 0.5
                A[i, i + 1] = 0.5

        A[-1, -1] = 1

        feas = collect_fea
        len_feas = []
        for fea in feas:
                len_feas.append(np.shape(fea)[0])

        _,D = np.shape(feas[0])
        hmm_means = np.zeros([N_state, N_mix, D])
        hmm_sigmas = np.zeros([N_state, N_mix, D])
        hmm_ws = np.zeros([N_state, N_mix])

        for s in range(N_state):
                sub_fea_collect = []
                # 初始化时 先为每个状态平均分配特征
                for fea, T in zip(feas, len_feas):
                        T_s = int(T / N_state) * s
                        T_e = (int(T / N_state)) * (s + 1)
                        sub_fea_collect.append(fea[T_s:T_e])

                ws,mus,sigmas=gen_para_GMM(sub_fea_collect, N_mix)
                hmm_means[s] = mus
                hmm_sigmas[s]=sigmas
                hmm_ws[s] = ws
        return pi,A,hmm_means, hmm_sigmas,hmm_ws
```

3.3 K-Means聚类

```
def run_kmeans(dataset, K, m = 20):
        labs = KMeans(n_clusters=K, random_state=9).fit_predict(dataset)
        return labs
```

3.4 生成 GMM 参数

```
def gen_para_GMM(fea_collect, N_mix):
        # 首先对特征进行kmeans聚类
        feas = np.concatenate(fea_collect, axis=0)
        N,D=np.shape(feas)
```

```
        labs = run_kmeans(feas, N_mix, m = 20)
        mus = np.zeros([N_mix, D])
        sigmas = np.zeros([N_mix, D])
        ws = np.zeros(N_mix)
        for m in range(N_mix):
                index = np.where(labs == m)[0]
                sub_feas = feas[index]
                mu = np.mean(sub_feas, axis = 0)
                sigma = np.var(sub_feas, axis = 0)
                sigma = sigma + 0.0001
                mus[m] = mu
                sigmas[m] = sigma
                ws[m] = np.shape(index)[0] / N
        ws = (ws + 0.01) / np.sum(ws + 0.01)
        return ws, mus, sigmas
```

3.5 整体训练流程

```
if __name__=="__main__":
        models = []
        train_path = "D://dev//code//python//data4.3//data4.3//train"
        for i in range(1,15):
                wav_path = os.path.join(train_path, str(i))
                collect_fea = []
                len_feas = []
                dirs = os.listdir(wav_path)
                for file in dirs:
                        if file.split(".")[-1]=="wav":
                                wav_file=os.path.join(wav_path, file)
                                fea=extract_MFCC(wav_file)
                                collect_fea.append(fea)
                                len_feas.append(np.shape(fea)[0])
                N_state = 4
                N_mix = 3
                pi, A, hmm_means, hmm_sigmas,
hmm_ws=init_parm_hmm(collect_fea,N_state, N_mix)
                train_GMMHMM=GMMHMM(n_components=N_state,
                                    n_mix=N_mix,
                                    covariance_type='diag',
                                    n_iter=90,
                                    tol=1e-5,
                                    verbose=False,
                                    params="tmcw",
                                    min_covar=0.0001
                                    )
                train_GMMHMM.startprob_=pi
                train_GMMHMM.transmat_=A

                train_GMMHMM.weights_=hmm_ws
                train_GMMHMM.means_=hmm_means
                train_GMMHMM.covars_=hmm_sigmas

                print("train GMM-HMM",i)
                train_GMMHMM.fit(np.concatenate(collect_fea, axis = 0),
np.array(len_feas))
```

```
                models.append(train_GMMHMM)
        np.save("models hmmlearn.npy", models)
```

3.6 测试

```
        test_dir = "D://dev//code//python//data4.3//data4.3//test"
        models = np.load("models hmmlearn.npy", allow_pickle=True)
        count = 0
        count2 = 0

        for i in range(98):
            wav_file = os.path.join(test_dir, str(i + 1) + ".wav")
            fea = extract_MFCC(wav_file)
            lab_true = (int)(i // 7) + 1
            scores = []
            scores2 = []
            for m in range(1, 15):
                    model = models[m - 1]
                    score, _ = model.decode(fea)
                    scores.append(score)
                    score2 = model.score(fea)
                    scores2.append(score2)

            det_lab = np.argmax(scores) + 1
            det_lab2 = np.argmax(scores2) + 1
            if det_lab == lab_true:
                count = count + 1
            if det_lab2 == lab_true:
                count2 = count2 + 1

            print("true lab %d det lab1 %d det lab2 %d "%(lab_true, det_lab,
det_lab2))

        print("decode %.2f   "%(count*100/98))
```

```
true lab 13 det lab1 13 det lab2 13
true lab 13 det lab1 13 det lab2 13
true lab 14 det lab1 14 det lab2 14
true lab 14 det lab1 14 det lab2 14
true lab 14 det lab1 14 det lab2 14
true lab 14 det lab1 14 det lab2 14
true lab 14 det lab1 14 det lab2 14
true lab 14 det lab1 14 det lab2 14
true lab 14 det lab1 14 det lab2 14
decode 100.00
```

4 实验过程 连续词

4.1 导入所需库 配置路径

```
import os
import pickle
import numpy as np
import scipy.io.wavfile as wvf
from python_speech_features import mfcc
from hmmlearn.hmm import GMMHMM
import heapq
import logging

# 训练数据路径
data_path="D://dev//code//python//data4//data4"
# 模型保存路径
model_path = "hmm_gmm_model.pkl"
```

4.2 特征提取函数

```
def wav2mfcc(label, data_path):
    trng_data = {}
    write_pickle = True
    mfccs = []
    rate, sig = wvf.read(data_path)
    mfcc_feat = mfcc(sig, rate)
    mfccs.append(mfcc_feat)
    trng_data[label] = mfccs
    return trng_data
```

4.3 高斯混合模型配置信息

```
def obtain_config(label):
    conf = {}
    conf[label] = {}
    conf[label]["n_components"] = 2
    conf[label]["n_mix"] = 2
    return conf
```

4.4 创建GMMHMM模型

```
def get_hmm_gmm(label, trng_data = None, GMM_config=None,
model_path="hmm_gmm_model.pkl", from_file=False):
    hmm_gmm = {}
    if not from_file:
        hmm_gmm[label] = GMMHMM(
            n_components=GMM_config[label]["n_components"],
            n_mix=GMM_config[label]["n_mix"]
        )
        if trng_data[label]:
            hmm_gmm[label].fit(np.vstack(trng_data[label]))
        pickle.dump(hmm_gmm,open(model_path, "wb"))
    else:
        hmm_gmm = pickle.load(open(model_path, "rb"))
    return hmm_gmm
```

4.5 读取训练数据 训练模型

```
def train(data_path, model_path):
    with open(os.path.join(data_path, "label.txt")) as f:
        label = f.readline()
    data_path = os.path.join(data_path, "train.wav")
    train_data = wav2mfcc(label, data_path)
    GMM_config = obtain_config(label)
    hmm_gmm = get_hmm_gmm(label, train_data, GMM_config, model_path)
    return hmm_gmm

hmm_gmm = train(data_path, model_path)
```

4.6 测试

```
def test_file(test_file, hmm_gmm):
    rate, sig = wvf.read(test_file)
    mfcc_feat = mfcc(sig, rate)
    pred = {}
    for model in hmm_gmm:
        pred[model] = hmm_gmm[model].score(mfcc_feat)
    return get_nbest(pred, 2), pred

def get_nbest(d, n):
    return heapq.nlargest(n, d, key = lambda k : d[k])
def predict_label(file, hmm_gmm):
    predicted = test_file(file, hmm_gmm)
    return predicted
logging.getLogger("hmmlearn").setLevel("CRITICAL")

wave_path = os.path.join(data_path, "train.wav")
predicted, probs = predict_label(wave_path, hmm_gmm)

print("PREDICITED : %s" % predicted[0])
```

```
[Running] set PYTHONIOENCODING=utf8 && python "d:\dev\code\python\4_3_2.py"
PREDICITED : i like you , do you like me
```

5 实验总结

本实验的目标是使用高斯混合模型隐马尔可夫模型（GMM-HMM）对音频数据进行特征提取和模式识别。实验涉及训练GMM-HMM模型，使用模型对音频文件进行分类，并测试模型的性能。GMM-HMM模型是处理音频分类任务的有效工具。通过适当的特征提取和模型训练，可以实现对音频数据的有效识别和分类。实验还展示了如何使用Python的相关库来实现这一过程，包括音频处理、特征提取、模型训练、保存、加载和预测。进一步的工作可以探索更多特征提取技术、优化模型参数，以及使用更大的数据集来提高模型的准确性和鲁棒性。同时，可以探索将GMM-HMM与其他机器学习或深度学习方法结合使用，以进一步提高音频分类的性能。