# 创建一个语音识别器

- import os
- import argparse
- import numpy as np
- from scipy.io import wavfile
- from hmmlearn import hmm   # 要去网址下载whl包直接安装from python_speech_features import mfcc
- '''   功能作用 : 创建一个语音识别器   创建隐马尔科夫模型   注释 : 将用到隐马尔科夫模型（Hidden Markov Models，HMMs）来做语音识别。隐马尔科夫模型非常擅长建立时间序列数据模型。因为一个音频信号同时也是一个时间序列信号，因此隐马尔科夫模型也同样适用于音频信号的处理。假定输出是通过隐藏状态生成的，我们的目标是找到这些隐藏状态，以便对信号建模'''
- # 函数的作用是 : 解析输入的参数
- def build_arg_parser():
    - parser = argparse.ArgumentParser(description='Trains the HMM classifier')
    - parser.add_argument("--input-folder", dest="input_folder", required=True, help="Input folder containing the audio files in subfolders")
    - return parser
- # 创建类处理HMM相关过程class HMMTrainer(object):
- '''   参数 : n_components定义了隐藏状态的个数   参数 : cov_type定义了转移矩阵的协方差类型   参数 : n_iter定义了训练的迭代次数   '''
- def __init__(self, model_name='GaussianHMM', n_components=4, cov_type='diag', n_iter=1000):
    - self.model_name = model_name
    - self.n_components = n_components
    - self.cov_type = cov_type
    - self.n_iter = n_iter
    - self.models = []
    - if self.model_name == 'GaussianHMM':
    - self.model = hmm.GaussianHMM(n_components=self.n_components, covariance_type=self.cov_type, n_iter=self.n_iter)
    - else:
    - raise TypeError('Invalid model type')
- # X是二维数组，其中每一行是13维   def train(self, X):
- np.seterr(all='ignore')
- self.models.append(self.model.fit(X))
- # 基于该模型定义一个提取分数的方法 ,对输入数据运行模型
- def get_score(self, input_data):
    - return self.model.score(input_data)
- if __name__=='__main__':
    - args = build_arg_parser().parse_args()

  - input_folder = args.input_folder
- # 初始化隐马尔科夫模型的变量
  - hmm_models = []
- # 解析输入路径
- for dirname in os.listdir(input_folder):
- # 获取子文件夹名称
- subfolder = os.path.join(input_folder, dirname)
- if not os.path.isdir(subfolder):
  - continue
- # 提取标记        label = subfolder[subfolder.rfind('/') + 1:]
- # 初始化变量
- X = np.array([])
- y_words = []
- # 迭代所有的音频文件 (分别保留一个进行测试)
- for filename in [x for x in os.listdir(subfolder) if x.endswith('.wav')][:-1]:
- # 读取一个音频文件
-  filepath = os.path.join(subfolder, filename)
- sampling_freq, audio = wavfile.read(filepath)
- # 提取 mfcc 的特征
- mfcc_features = mfcc(audio, sampling_freq)
- # 将mfcc特征添加到X变量
-   if len(X) == 0:
  - X = mfcc_features
  - else:
  - X = np.append(X, mfcc_features, axis=0)
- # 添加标记
- y_words.append(label)
  - print ('X.shape =', X.shape)
- # 训练并保存HMM模型
  - hmm_trainer = HMMTrainer()
  - hmm_trainer.train(X)
  - hmm_models.append((hmm_trainer, label))
  - hmm_trainer = None    # 测试文件
  -  input_files = [
    - 'data/pineapple/pineapple15.wav',
    - 'data/orange/orange15.wav',
    -  'data/apple/apple15.wav',
    -  'data/kiwi/kiwi15.wav'           ]
- # 为输入数据分类
- for input_file in input_files:
- # 读取每一个音频文件
- sampling_freq, audio = wavfile.read(input_file)

- # 提取 mfcc 特征
- mfcc_features = mfcc(audio, sampling_freq)
- # 定义变量
- max_score = None
- output_label = None
- # 迭代HMM模型并选取得分最高的模型
- for item in hmm_models:
  - hmm_model, label = item
- # 提取分数，并保存最大分数值
- score = hmm_model.get_score(mfcc_features)
- if score > max_score:
- max_score = score
- output_label = label
- # 打印结果
- print ("\nTrue:", input_file[input_file.find('/')+1:input_file.rfind('/')])
- print ("Predicted:", output_label )