assignment_9_2

September 25, 2022

[1]:
```
import os
import numpy as np
import pickle
import glob
import librosa
from pydub import AudioSegment
from pydub.utils import mediainfo
from sklearn import preprocessing
from sklearn.mixture import GaussianMixture
from sklearn import preprocessing
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
from tqdm import tqdm
import warnings
warnings.filterwarnings("ignore")
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning:
scipy._lib.messagestream.MessageStream size changed, may indicate binary
incompatibility. Expected 56 from C header, got 64 from PyObject
C:\Users\singh\anaconda3\lib\site-packages\pydub\utils.py:170: RuntimeWarning:
Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not work
  warn("Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not
work", RuntimeWarning)
```

[2]:
```
def mfcc_extraction(audio_filename, #.wav filename
                    hop_duration, #hop_length in seconds, e.g., 0.015s (i.e.,␣
 ↪15ms)
                    num_mfcc #number of mfcc features
                   ):
    speech = AudioSegment.from_wav(audio_filename) #Read audio data from file
    samples = speech.get_array_of_samples() #samples x(t)
    sampling_rate = speech.frame_rate #sampling rate f
    mfcc = librosa.feature.mfcc(
    y = np.float32(samples),
    sr = sampling_rate,
    hop_length = int(sampling_rate * hop_duration),
    n_mfcc = num_mfcc)
    return mfcc.T
```

[3]:
```
def learningGMM(features, #list of feature vectors, each feature vector is an␣
  ↪array
                n_components, #the number of components
                max_iter #maximum number of iterations
               ):
    gmm = GaussianMixture(n_components = n_components, max_iter = max_iter)
    gmm.fit(features)
    return gmm
```

[4]:
```
path = 'SpeakerData/'
speakers = os.listdir(path + 'Train/')
print(speakers)
```

```
['Anthony', 'AppleEater', 'Ara', 'Argail', 'Ariyan', 'Arjuan', 'Artem',
'Arthur', 'Artk', 'Arun', 'Arvala', 'Asalkeld', 'Asladic', 'Asp', 'Azmisov',
'B', 'Bachroxx', 'Bae', 'Bahoke', 'Bareford', 'Bart', 'Bassel', 'Beady', 'Beez',
'BelmontGuy']
```

[5]:
```
#this list is used to store the MFCC features of all training data of all␣
  ↪speakers
mfcc_all_speakers = []
hop_duration = 0.015 #15ms
num_mfcc = 12
for s in speakers:
    sub_path = path + 'Train/' + s + '/'
    sub_file_names = [os.path.join(sub_path, f) for f in os.listdir(sub_path)]
    mfcc_one_speaker = np.asarray(())
    for fn in sub_file_names:
        mfcc_one_file = mfcc_extraction(fn, hop_duration, num_mfcc)
        if mfcc_one_speaker.size == 0:
            mfcc_one_speaker = mfcc_one_file
        else:
            mfcc_one_speaker = np.vstack((mfcc_one_speaker, mfcc_one_file))
    mfcc_all_speakers.append(mfcc_one_speaker)
```

[6]:
```
for i in range(0, len(speakers)):
    with open('TrainingFeatures/' + speakers[i] + '_mfcc.fea','wb') as f:
        pickle.dump(mfcc_all_speakers[i], f)
```

[7]:
```
n_components = 5
max_iter = 50
gmms = [] #list of GMMs, each is for a speaker
for i in range(0, len(speakers)):
    gmm = learningGMM(mfcc_all_speakers[i],
                      n_components,
                      max_iter)
    gmms.append(gmm)
```

[8]:

```
for i in range(len(speakers)):
    with open('Models/' + speakers[i] + '.gmm', 'wb') as f: #'wb' is for binary␣
 ↪write
        pickle.dump(gmms[i], f)
```

[9]:

```
gmms = []
for i in range(len(speakers)):
    with open('Models/' + speakers[i] + '.gmm', 'rb') as f: #'wb' is for binary␣
 ↪write
        gmm = pickle.load(f)
        gmms.append(gmm)
```

[10]:

```
hop_duration = 0.015 #15ms
num_mfcc = 12
def speaker_recognition(audio_file_name, gmms):
    speaker_id = 0 #you need to calculate this
    score_list = []
    for i, gmm in enumerate(gmms):
        mfcc_test = mfcc_extraction(audio_file_name, hop_duration, num_mfcc)
        score_list.append((i,gmms[i].score(mfcc_test)))
    max_element = max(score_list, key=lambda x:x[1])
    #print("Max: ", max_element)
    #print("Max Index: ", max_element[0])
    speaker_id = max_element[0]
    return speaker_id
```

[11]:

```
speaker_id = speaker_recognition('SpeakerData/Test/Ara/a0522.wav', gmms)
print(speakers[speaker_id])
```

```
Ara
```

[12]:

```
files = glob.glob("SpeakerData/Test/*/*")
y_true = []
y_pred = []
for file in tqdm(files):
    #print ("Processing file:", file)
    #print (os.path.basename(os.path.dirname(file)))
    true_label = os.path.basename(os.path.dirname(file))
    speaker_id = speaker_recognition(file, gmms)
    pred_label = speakers[speaker_id]
    y_true.append(true_label)
    y_pred.append(pred_label)
    #print("True: ", true_label, " Pred:", pred_label)
```

```
100%|

        | 175/175 [01:33<00:00,  1.88it/s]
```

[13]:

```python
cm = confusion_matrix(y_true, y_pred)
print("Classification Report\n", classification_report(y_true, y_pred))
print("Confusion Matrix\n", cm)
print("Overall Accuracy:\n", accuracy_score(y_true, y_pred))
```

```
Classification Report
               precision    recall  f1-score   support

     Anthony       1.00      0.14      0.25         7
  AppleEater       1.00      1.00      1.00         7
         Ara       1.00      1.00      1.00         7
      Argail       1.00      1.00      1.00         7
      Ariyan       1.00      1.00      1.00         7
      Arjuan       1.00      1.00      1.00         7
       Artem       1.00      1.00      1.00         7
      Arthur       0.39      1.00      0.56         7
        Artk       1.00      1.00      1.00         7
        Arun       1.00      1.00      1.00         7
      Arvala       1.00      1.00      1.00         7
    Asalkeld       1.00      1.00      1.00         7
     Asladic       1.00      1.00      1.00         7
         Asp       1.00      1.00      1.00         7
     Azmisov       1.00      1.00      1.00         7
           B       1.00      1.00      1.00         7
    Bachroxx       1.00      1.00      1.00         7
         Bae       1.00      1.00      1.00         7
      Bahoke       1.00      0.86      0.92         7
    Bareford       1.00      1.00      1.00         7
        Bart       1.00      0.29      0.44         7
      Bassel       0.88      1.00      0.93         7
       Beady       1.00      1.00      1.00         7
        Beez       1.00      1.00      1.00         7
  BelmontGuy       1.00      1.00      1.00         7

    accuracy                           0.93       175
   macro avg       0.97      0.93      0.92       175
weighted avg       0.97      0.93      0.92       175

Confusion Matrix
 [[1 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0]
 [0 0 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7]]
Overall Accuracy:
 0.9314285714285714
```

```
[ ]:
```