基于视频信息的说话人识别与分离

秦圣岭 2019011076　　彭熙然 2019011183　　邢格格 2019011123

1. 问题背景

生活中时常会有多人同时讲话的场景。从一段嘈杂的录音中准确捕捉到每个说话人的音频可能有些困难，但当面对面交谈时，往往可以容易地区分，特别是当我们对说话人的身份很熟悉时。这说明视觉信息对于语音的分离是有益的。

本次大作业逐步实现基于单人说话视频中视觉和声音信息的说话人识别，以及对多音源视频的逐说话人音源分离。我们采用的方法有如下优势：

1. 方法简单高效，准确率很高，不使用深度学习方法，全部使用 CPU 计算，完全不需要 pytorch/tensorflow 和 GPU；

2. 调用的外部库很少，只使用了 dlib、sklearn、python_speech_features 等用于提取特征；

3. 运行时间非常短，运行一次测试只需要约半分钟，Task1 和 Task2 在 5 秒左右就能完成测试，Task3 也只需要 20 秒左右。

2. 实验内容

2.1. 基于视觉信息的说话人识别

Task1 要求对数据中的每一个无声音单人说话视频，通过与训练集中 20 个说话人视觉信息进行模板匹配或其它匹配方法，判断出对应说话人的 ID，实现仅根据视觉信息对说话人进行识别。

2.1.1　原理

Task1 使用 dlib 库中的人脸特征提取函数实现。该函数基于训练好的 ResNet 模型，可以快速检测人脸的 68 个关键点，提取出 128 维的特征。训练时，提取每个说话人的特征建立人脸特征库，测试时，只需要再提取样本的 128 维特征，寻找特征库中与其距离最近的特征向量，就可以识别出说话人。

识别过程流程如图1所示。

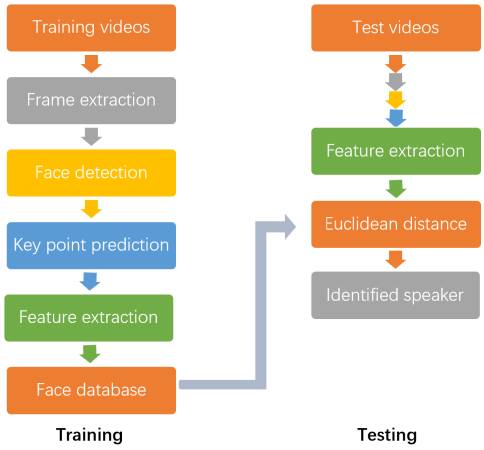

图 1. Task1 流程图

2.1.2　实现方法

Task1 仅额外依赖于计算机视觉库 dlib，其中的 shape_predictor() 可以确定人脸位置，face_recognition_model_v1() 可以检测人脸 68 特征点检测器，并将此 68 个关键点转换为 128 维特征。最终测试时，只需要将视频抽帧，计算待检测人与各个 ID 的欧氏距离，取距离最小时对应的 ID 作为最终结果即可。

关键代码如图2, 图3所示。

2.1.3　结果展示及分析

Task1 的准确率为 94%(图4)，整体准确率符合要求。其中，测试视频 024.mp4 真实编号为 ID6，错判成 ID5;034.mp4 真实编号为 ID19，错判成 ID9;046.mp4

```python
def get_feature(img):
    dets = detector(img)
    if len(dets):
        d = dets[0]
        l = np.maximum(d.left(), 0)
        u = np.maximum(d.top(), 0)
        r = np.minimum(d.right(), img.shape[1])
        d = np.minimum(d.bottom(), img.shape[0])
        rec = dlib.rectangle(l, u, r, d)
        shape = predictor(img, rec)

        descriptor = facerec.compute_face_descriptor(img, shape)
        feature = np.array(descriptor).reshape((1, 128))

        return feature
    else:
        return -1
```

图 2. 特征提取函数

```python
def test(feature_dict, video):
    diff = np.zeros((feature_dict.shape[0]))
    img = video[0]
    feature = get_feature(img)
    for i in range(feature_dict.shape[0]):
        diff[i] = np.linalg.norm(feature_dict[i] - feature)

    pred = np.argmin(diff) + 1
    return pred
```

图 3. 测试函数

```
Task 1 is running...
accuracy for task1 is: 0.94
```

图 4. 任务一结果

真实编号为 ID18，错判成 ID9。观察发现，ID5 和 ID6 错判可能由于二者均为西方女性，且相貌相似度较高，128D 特征值也较接近。ID9 为女性，但 ID18 和 19 均为长发男性，错判可能是三人面中部（鼻子及周围附近）较相似，且测试集中 ID19 面部遮挡较多（帽子、墨镜），ID18 图像明暗对比度较低等原因使 128D 特征值不够个性化导致的。总体来说，结果比较理想。

2.2. 基于声音信息的说话人识别

任务二要求对数据中的每一个单声道单人说话音频，通过与训练集中 20 个说话人声音信息进行匹配，判断出对应说话人的 ID，实现仅根据声音信息对说话人进行识别。

2.2.1 原理

课程中讲过，梅尔谱是语音的重要特征。本实验采用梅尔倒谱系数 (MFCC) 提取特征，并用高斯混合模型 (GMM) 聚类。首先提取演讲者训练样本的 20 维 MFCC 特征和 20 维 deltaMFCC 特征，将总共 40 维特征作为输入训练 GMM 模型。测试时，提取测试样本的 40 维特征，使用 GMM 模型来计算所有模型的特征分数，具有最高分数的说话人即为识别结果。

识别过程流程如图5所示。

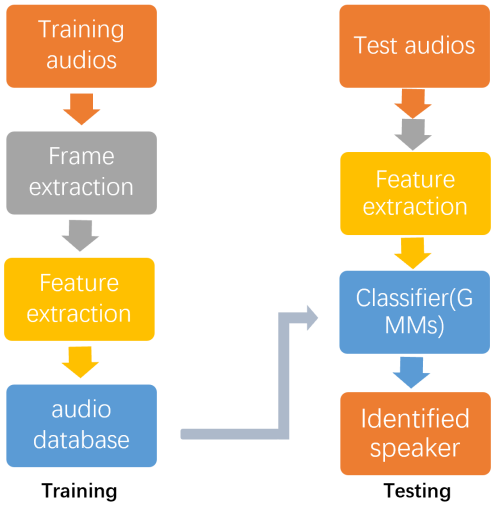

图 5. task2 流程图

2.2.2 实现方法

40 维特征提取 使用 python_speech_features 库从语音帧中提取 40 维特征，包括 20 个 MFCC 特征和 20 个 MFCC 的 2 阶差分特征。MFCC 的差分能够提供 MMCC 一段时间内的动态信息。为计算 MFCC 的增量特征，我们应用以下等式：

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

N 为总增量数。

关键代码如图6所示。

```python
def get_feature(audio, fs):
    mfcc_feat = mfcc(audio, fs, 0.025, 0.01, 20, appendEnergy=True)
    mfcc_feat = preprocessing.scale(mfcc_feat)
    delta_feat = calculate_delta(mfcc_feat)
    feature = np.hstack((mfcc_feat, delta_feat))
    return feature
```

图 6. 特征提取函数

训练模型 首先需要运用 GMM 对所有说话人进行独立建模。通过高斯分布/聚类的线性组合来近似类的概率分布，即 GMM 的分量。模型的数据点（特征向量）的可能性可以通过以下等式计算：

$$P(X|\lambda) = \sum_{k=1}^{K} \omega_k P_k(X|\mu_k, \Sigma_k)$$

其中 $P_k(X|\mu_k, \Sigma_k)$ 为高斯分布：

$$P_k(X|\mu_k, \Sigma_k) = \frac{1}{\sqrt{2\pi|\Sigma_k|}} e^{\frac{1}{2}(X-\mu k)^T \sum^{-1}(X-\mu_k)}$$

训练参数 X_i 用于估计 k 分量的均值 μ、方差 Σ 和权重 ω，通过 K 均值算法识别数据中的 k 个聚类，并为每个聚类分配相等的权重 $\omega = \frac{1}{k}$。然后将'k' 高斯分布拟合到这些聚类，采用最大期望 (EM) 算法使所有参数迭代更新，直到收敛。

测试 首先提取测试样本的 40 维特征，计算对数似然分数 $P(x_i|S_j)$。每一帧来自特定说话人的可能性通过替换似然方程中 GMM 模型的 μ 和 Σ 参数计算得到。从这些分量中取 $k\omega$ 似然的加权和。对样本的所有帧重复此操作，累计具有最高似然分数的说话人模型被视为已识别的说话人。

关键代码如图7所示。

2.2.3 结果展示及分析

Task2 的准确率为 88%(图8)，稍微不太理想。注意到很多说话样本中有多人一起说话，导致提取的特征有部分混叠，是结果不够理想的原因之一。

2.3. 双声道多音源视频的逐说话人音源分离

通过测试视频每个说话人对应区域进行说话人识别，进而借助说话人信息对测试视频进行音源分离。任务三要求对数据中的每一段双声道多音源视频，根据左中右 3 个说话人的信息，按顺序分离出每个说话人的音频，实现结合音视频信息分离多人同时讲话时的音源。

```python
def test(audio_trace):
    source_path = 'lib/model.out'
    models = pickle.load(open(source_path, 'rb'))
    speakers = [x for x in range(1, 21)]
    vec = get_feature(audio_trace, 16000)

    log_likelihood = np.zeros(len(models))

    for i in range(len(models)):
        gmm = models[i]
        scores = np.array(gmm.score(vec))
        log_likelihood[i] = scores.sum()

    winner = np.argmax(log_likelihood)
    ID = speakers[winner]

    return ID
```

图 7. 测试函数

```
Task 2 is running...
accuracy for task2 is: 0.88
```

图 8. 任务二结果

2.3.1 原理

本实验采用独立成分分析（ICA）将双声道音频盲分离，然后根据任务一和任务二已得的模型进行匹配，即可匹配音频与说话人位置。识别过程流程如图9所示。

2.3.2 实现方法

在基本的 ICA 中，应有 n 个信号源和 n 个接收器，设得到 m 组采样，则得到的数据是一个 $n \times m$ 的矩阵，$X_{n\times m} = [x^{(1)} \quad x^{(2)} \quad \ldots\ldots \quad x^{(m)}]$，源信号为 $S_{n\times m} = [s^{(1)} \quad s^{(2)} \quad \ldots\ldots \quad s^{(m)}]$。设存在一混淆矩阵 $A_{n\times n}$ 使下式成立：

$$X_{n\times m} = A_{n\times n} \cdot S_{n\times m}$$

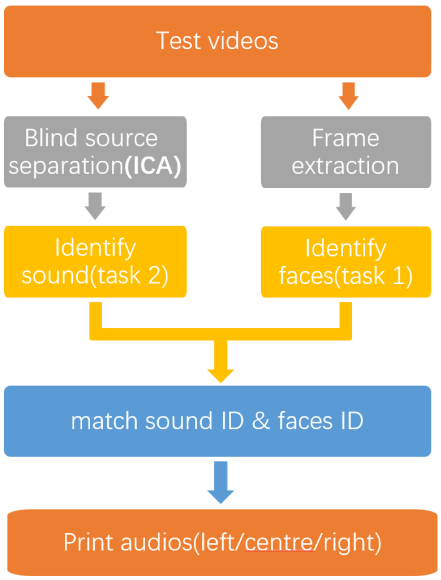

图 9. task3 流程图

则找到该混淆矩阵 $A_{n\times n}$ 即可分离出源信号。

本实验用双声道采集了三个人的语音数据，等价于求解欠定方程，无法直接使用 ICA 分离出三个独立音频。但是，由于有两个观测值，可以使用 ICA 分离出两个独立音频。通过将混合音频向两个独立音频投影，估计出两个独立音频的幅度，然后用混合音频信号减去两个独立音频信号，即可得到第三个音频。

使用 sklearn 中的 FastICA() 函数分离出两段音频，投影过程可以直接用 numpy 实现。

关键代码如图10所示。

```
def task3(video_frames, audio_trace):
    ica = FastICA(n_components=2, max_iter=300)
    separated = ica.fit_transform(audio_trace)
    rho_0 = np.sum(audio_trace[:, 0] * separated[:, 0]) / np.linalg.norm(separated[:, 0]) / np.linalg.norm(
        audio_trace[:, 0])
    rho_1 = np.sum(audio_trace[:, 0] * separated[:, 1]) / np.linalg.norm(separated[:, 1]) / np.linalg.norm(
        audio_trace[:, 1])
    audio_third = audio_trace[:, 0] - separated[:, 0] * rho_0 - separated[:, 1] * rho_1
    separated = np.column_stack((separated, audio_third))
    separated = np.transpose(separated)
    posed = pos(video_frames, separated)

    return posed
```

图 10. FastICA

2.3.3 结果展示及分析

双声道盲分离结果较好，匹配分离的 SISDR 效果稍微差一些。由于音频识别可能出错，会降低匹配分离的 SISDR。

```
Task 3 is running...
strength-averaged SISDR_blind for task3 is: 7.576322487200551
strength-averaged SISDR_match for task3 is:  5.642772654830838
```

图 11. 任务三结果

* 注：将 ICA 算法应用在双声道三音源会具有一定的随机性，即分离出的独立信号幅度和顺序无法从内部确定，因此实际测试结果可能具有不稳定性，多运行几次可以获得较为理想的结果。

3. 总结

本次大作业分为三个子任务，从音频、视频、音视频结合三种角度层次递进，将课上学习和查阅资料学习的视听信号信息处理技术用于说话人识别与分离的实际场景，是一次有趣有挑战性的尝试。我们保证我们的结果能简单复现。很高兴能用非深度学习的方法较好地完成任务，方法基本来自课堂，感谢老师和助教的付出！

附录

A 代码清单

./xxx	根目录
| test.py	函数入口
| task1.py	完成 Task1
| task2.py	完成 Task2
| task3.py	完成 Task3
| ./lib	依赖库文件夹
| | feature_dict.npy	人脸特征库
| | model.out	音频特征库
| | dlib_face_recognition_resnet_model_v1.dat	人脸特征提取依赖库
| | shape_predictor_68_face_landmarks.dat	人脸位置识别依赖库
| ./train	训练数据
| ./test_offline	线下测试数据
| ./test_online	线上测试数据，不公开

B 运行方式

1. 按上文方式组织文件；
2. conda create -n vasp python=3.6；
3. pip install -r requirements.txt –ignore-installed (需要安装 cmake)；
4. python test.py.

参考文献

[1]facerec = dlib.face_recognition_model_v1() 面部识别器用法
https://blog.csdn.net/weixin_44493841/article/details/93504996

[2]Spoken Speaker Identification based on Gaussian Mixture Models : Python Implementation
https://appliedmachinelearning.blog/

[3] 独立成分分析 (ICA, FastICA) 及应用
https://blog.csdn.net/ctyqy2015301200079/article/details/86705869

[4] 因子分析、主成分分析、独立成分分析——斯坦福 CS229 机器学习个人总结
https://blog.csdn.net/sinat_37965706/article/details/71330979