基于 x-vector 的说话人识别系统的复现

黄昱斌

摘要

说话人识别技术经历了高斯混合模型-通用背景模型（GMM-UBM）、高斯混合模型-支持向量机（GMM-SVM）、联合因子分析（JFA）、i-vector 等传统方法识别阶段后，进入了利用深度学习实现说话人识别的阶段。本文基于论文 *X-VECTORS: ROBUST DNN EMBEDDINGS FOR SPEAKER RECOGNITION*，利用 Kaldi 工具在 Aishell-1 数据库上实现了基于 x-vector 的说话人识别，并在最后利用 EER 与 minDCF 两个指标评价复现工作的性能。

关键词：说话人识别；Kaldi；x-vector

1 引言

说话人识别（Speaker Recognition）是指利用语音这一生物特征让计算机实现一定的判断的过程，它包含了说话人辨认（Speaker Identification）和说话人确认（Speaker Verification）。前者是要判断待识别语音是由给定说话人集合中的哪一位所说（可看作是一个多分类问题），而后者是要判断待识别语音是否是指定的某个人所说。相比于深度学习下的说话人识别技术，传统的方法结构一般属于浅层结构，只是对原始的输入信号进行较少层次的线性或者非线性处理以达到信号与信息处理的目的。深层次的网络模型更能捕捉到更多的特征，以实现更好的实验效果[1]。论文 *X-VECTORS: ROBUST DNN EMBEDDINGS FOR SPEAKER RECOGNITION*[2]为我们说明了实验的全部流程，为了模拟论文中的实验，我们利用 Kaldi 工具在 Aishell-1 数据集上实现了基于 x-vector 的说话人识别。

2 相关工作

2.1 传统的基于 i-vector 的说话人识别方法

基于 i-vector 的说话人识别方法是传统方法中较为经典的一种，它不包含深度神经网络。在 kaldi 工具包中就是使用的 i-vector 与 PLDA 的方法进行的说话人识别，代码位置在 kaldi/egs/aishell/v1。它的流程为：数据准备、提取 MFCC 特征、训练 UBM、训练并提取 i-vector、训练 PLDA 打分器以及用测试集来检验。i-vector 本质上也是表征说话人特征的高维向量，在[3]中就早已提出了这种方法，它综合考虑了说话人信号和信道信号的因素。

2.2 基于 x-vector 的说话人识别方法

随着技术的发展和日渐成熟，深度学习也逐渐进入了语音处理领域，本次拟复现的论文就是利用深度神经网络的特性，来提取出更加贴近真实说话人的说话人嵌入向量。在深度学习的环境下，我们需要大量的说话人语音数据，并将他们划分为训练集和测试集。训练集用于训练神经网络以更准确地去提取说话人表征向量，还用于训练 PLDA 打分器用于识别完成后的相似性判别。这部分上游工作花费的时间也较长。测试集划分为注册集和评估集，注册集用于在于先让系统知道所有待识别的对象，评估集则是用于最后的说话人识别。于前一种方法不同的地方主要在于采用了深度神经网络，能够提取到更多的特征以取得更好的效果。

3 本文方法

3.1 本文方法概述

本次复现利用了深度神经网络来处理语音信号转化来的 MFCC 特征序列，通过 PLDA 打分器来判别两个向量的相似度，最终达到识别说话人的效果。

3.2 特征提取模块

此部分对论文中的深度神经网络进行说明，其网络架构如下图所示：

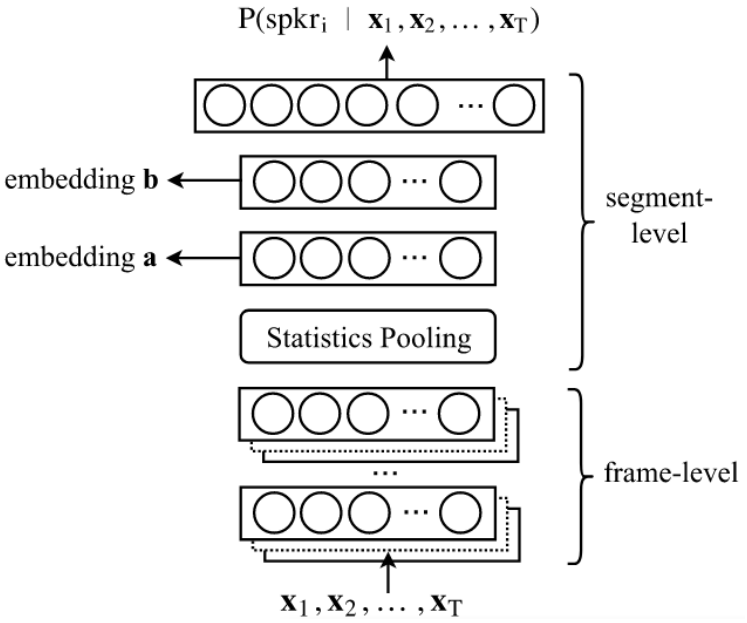

图 1: 方法示意图

该网络的输入是语音经过一系列预处理得到的 MFCC 特征序列，中间包含的结构有时延神经网络 TDNN、统计池化层、在段级操作的附加层以及最后输出的 softmax 层。

网络的前 5 层在帧级工作，采用的是时延神经网络 TDNN。假设 t 是当前时间步长，在第一层输入中，将这 5 帧 {t−2, t-1, t, t+1, t+2} 一起输入到下一层，接下来两层分别是 {t−2, t, t+2} 和 {t−3, t, t+3}，最后 2 层也在帧级工作但没有添加时间上下文。

统计池化层接收最终帧级层的输出作为输入，聚合输入段并计算其均值和标准差。这些段级统计数据连接在一起，并传递给两个额外的隐藏层，维度为 512 和 300（两者都可用于计算嵌入），最后是 softmax 输出层。排除 softmax 输出层（因为训练后不需要），总共有 4400 万个参数。

3.3 损失函数定义

作者采用的损失函数为多类交叉熵损失函数，其公式如下：

$$E = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk} ln(P(spkr_k \mid \mathbf{x}_{1:T}^{(n)}))$$

假设在 N 个训练语音段中有 K 个说话人，则条件概率部分的含义为给定语音帧 x1 xT 的条件下为说话人 k 的概率。如果前面系数 d 的含义为如果语音段 n 的说话人为 k，则 d 为 1，否则为 0。

4 复现细节

4.1 与已有开源代码对比

本次复现工作利用的开源代码仓库地址为 https://github.com/qiny1012/kaldi_x-vector_aishell.git，由于实验算力资源紧张，故在第六个脚本中设为不使用 GPU，实验共用时 4 天多的时间。改动的地方如下图：

```
steps/nnet3/train_raw_dnn.py --stage=$train_stage \
  --cmd="$train_cmd" \
  --trainer.optimization.proportional-shrink 10 \
  --trainer.optimization.momentum=0.5 \
  --trainer.optimization.num-jobs-initial=1 \
  --trainer.optimization.num-jobs-final=1 \
  --trainer.optimization.initial-effective-lrate=0.001 \
  --trainer.optimization.final-effective-lrate=0.0001 \
  --trainer.optimization.minibatch-size=64 \
  --dir=$nnet_dir \
  --trainer.srand=$srand \
  --trainer.max-param-change=2 \
  --trainer.num-epochs=80 \
  --trainer.dropout-schedule="$dropout_schedule" \
  --trainer.shuffle-buffer-size=1000 \
  --egs.frames-per-eg=1 \
  --egs.dir="$egs_dir" \
  --cleanup.remove-egs $remove_egs \
  --cleanup.preserve-model-interval=10 \
  --use-gpu=false || exit 1 ;
```

图 2: 改动的代码

4.2 实验环境搭建

第一步，从 Kaldi 的 github 仓库中 clone 下代码到自己的机器上，需要提前安装好 git 工具。使用的命令是 git clone https://github.com/kaldi-asr/kaldi，进度条完成后会在当前文件夹下生成一个名为“kaldi”的文件夹，里面包含了所有需要的工具、配置文件以及案例。

第二步，进入 kaldi/tools, 使用命令 extras/check_dependencies.sh。这条命令会帮助我们检查出当前我们还没有安装的工具和一些依赖，比如 vim、gcc。等这条命令执行完后就会显示我们目前缺少的东西，同时也会非常友好地给我们相应的安装命令, 照着输入就能够安装好了。最后再次输入 extras/check_dependencies.sh，当界面提示所有工具以及安装完成时，说明这一步工作已经到位, 最后输入 make -j 8 来编译刚才的环境即可。

第三步，进入 kaldi/src，输入./configure –shared ，来配置 src 文件夹中的内容，显示成功后，接着输入 make depend 和 make 两条命令来编译 src 文件夹，可以加入 -j 8 这个选项参数来加速编译进程。当界面没有报错时，编译即成功了。在这一步若遇到 gcc 版本不合适的情况，会在./configure –shared 命令后报错，提示需要特定的 gcc 版本。接下来需要做的就是安装相应版本的 gcc，若系统中已经有多个版本的 gcc，我们只需要修改软连接即可。

第四步，就是要检查之前所有的安装工作是否成功，最经典的例子就是使用 kaldi/egs/yesno/s5 这个例子来检验安装工作。我们进入这个文件夹中直接运行 run.sh 脚本，如果没有报错，则说明最基本的 Kaldi 实验环境已经成功安装。

4.3 具体实验流程

第一步，我们需要进行数据的预处理，音频数据不同于图像，能够在计算机中直接读取为数字信号。音频数据需要经过一系列操作转化为 MFCC 特征，即一个若干维的向量。本次实验中的神经网络

的输入就是 MFCC 特征序列。本次实验的预处理包含了数据准备、语音转化为 MFCC 特征、数据增广以及增广后数据转化为 MFCC 特征。数据准备即把数据规范成 Kaldi 规定的数据文件夹格式，划分训练集和测试集。语音转化为 MFCC 特征需要经过快速傅里叶变换、三角滤波、取对数、DTC 变换，得到的向量即上文提到的 MFCC 向量，这一步还需要经过 VAD（语音活动检测），即从声音信号流里识别和消除长时间的静音期，以免对识别造成干扰。数据增广即使用专门的噪声数据集与原数据进行混合，可以让神经网络更加鲁棒。因为真实数据中噪声是很常见的，加入噪声，对网络进行训练，可以让其更好地适应真实场景的应用。最后再提取增广后数据的 MFCC 特征即可。最后实验的源码中还加入了过滤语音这一步骤，去除了语音数量少于 8 的说话人。

第二步，建立并训练 x-vector 提取器。把之前提取到的语音的 MFCC 特征送入神经网络进行训练。神经网络的结构在上文已经提到过了，这里不再赘述。

第三步，利用上一步训练好的网络提取训练数据的 x-vector，将他们进行 LDA 降维后再训练 PLDA 打分器。这一步是最后说话人识别的关键。

第四步，将测试集划分为注册集与评估集，最后通过脚本计算出识别的 EER 和 minDCF 指标。

5 实验结果分析

本次复现跑通了代码 2 次，结果如下图所示：

```
(base) huangyb@dell-PowerEdge-T550:~/kaldi/egs/sre16/v2$ ./new_09_result.sh
EER: 2.473%
minDCF(p-target=0.01): 0.3047
minDCF(p-target=0.001): 0.4671
```

```
(base) huangyb@dell-PowerEdge-T550:~/kaldi/egs/sre16/v2$ ./new_09_result.sh
EER: 2.473%
minDCF(p-target=0.01): 0.3045
minDCF(p-target=0.001): 0.4678
```

图 3: 实验结果示意

采用 EER 等错误率和 minDCF 最小代价函数来衡量说话人识别的性能。EER 是分类任务中常用的指标，它是指错误拒绝率 FRR 与错误接受率 FAR 相等时的百分比，即 FRR=FAR=EER。而等错误率只是简单地让 FAR 与 FRR 相等，以 EER 为指标去优化系统并不能带来更好的应用效果，因为实际应用中两种错误分类的代价是不同的。比如把有病的人误诊为无病的人和把无病的人误诊为有病的人，前者的代价明显大于后者。于是采用了代价检测函数，其公式如下：

$$C_{DCF} = C_{Miss} \cdot P_{target} \cdot FRR + C_{FalseAlarm} \cdot (1 - P_{target}) \cdot FAR$$

P-target 为真实说话人的先验概率，(1 - P-target) 为冒名顶替者的先验概率，C-miss 为错误拒绝率的权重，C-falsealarm 为错误接收率的权重，当代价检测函数取到最小值时即我们需要的性能指标 minDCF。

6 总结与展望

通过本次复现工作，理解了说话人识别的大致流程，深刻体会到了深度学习在语音处理领域的强大。但也有一些不足需要提升，一是还需要更加深入地去掌握一些细节问题；二是能继续去尝试使用 GPU 去训练，与不使用 GPU 所花的时间对比；三是去尝试在原有代码基础上作出创新来提升说话人

识别的系统性能；四是去分析同样的数据集上为什么 x-vector 的效果比 i-vector 更差，与预期效果相反。

参考文献

[1] 曾春艳; 马超峰; 王志锋; 朱栋梁; 赵楠; 王娟; 刘聪; 深度学习框架下说话人识别研究综述[J]. 计算机工程与应用, 2020, 56: 8-16.

[2] SNYDER D, GARCIA-ROMERO D, SELL G, et al. X-Vectors: Robust DNN Embeddings for Speaker Recognition[C]//2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2018: 5329-5333. DOI: 10.1109/ICASSP.2018.8461375.

[3] DEHAK N, KENNY P, DEHAK R, et al. Front-End Factor Analysis for Speaker Verification[J]. IEEE Transactions on Audio, Speech, and Language Processing, 2011.