说话人自适应

2019 年 1 月 13 日

1 什么是说话人自适应

故事发生在2018年10月，一位印度学者来实验室访问，做了一场关于“如何检测假冒说话人”的报告。这位仁兄讲的神采飞扬，底下的学生们却面面相觑，一头雾水。原因倒不是讲座的内容有多么高深，而是这位的英语实在太有特色了，标准高清孟买腔，且娴熟轻快，对我们这种习惯了English或是Chinglish的听众来说，实在是反应不过来。

人尚如此，遑论机器。

研究者很早就知道，不同说话人的生理结构不同，可能造成非常大的发音差异性。因此，训练一个适合多说话人的语音识别系统（能常称为**说话人无关系统**）要比训练一个只给一个人用的系统（通常称为**说话人相关系统**）要困难的多。所以，早期的语音识别系统几乎都是说话人相关的，直到80年代以后，随着数据的积累和建模技术的改进（特别是统计模型的广泛应用），说话人无关的识别系统才开始普及。然而，说话人之间的差异总是存在的，一个对所有人“通用”的系统总不如一个对个人“定制”的系统更有效。我们当然希望识别系统可以对所有人都有不错的效果，但更重要的是对一些特定人（如我自己，或前面那位印度学者）识别的更好。这就要用到**说话人自适应**（Speaker Adaptation）技术。

说话人自适应技术的基本思路很简单：给定一个说话人无关的识别系统，基于某一目标说话人的若干数据，通过对该识别系统的某些部分进行合理调节，使得调节后的系统对目标说话人的性能更好。这里的“数据”既可以是语音数据，也可以是文本数据；要调整的部分既可以是声学特征提取，也可以是声学模型或语言模型。在绝大多数情况下，说话人自适应指的是对说话人声学特性的适应，因此主要是对特征提取和声学模型的修正和调节。关于对说话人在用词、造句等语言特性，一般不认为是个人的特异性，而是和说话人所处的应用

场景相关，因此通常称为**领域自适应**（Domain Adaptation）。关于说话人自适应和领域自适应的更多知识，可参考相关的综述文章[1]。

2 特征域自适应与声道长度规整

对说话人进行自适应的一个简单思路是对他们发出的声音进行调整，以适应说话人无关的通用系统。这种依说话人特性对语音信号进行调节的方式称为特征域自适应。声道长度规整（Vocal Tract Length Normalization, VTLN）[2]是一种典型的特征域自适应方法。

VTLN的基本思路来源于人类的发音机理。研究者发现，人们在发音时，声音的特性和声道的长短有很大关系，这一关系可形式化为在频谱上的形变。例如，对同样一句话，声道长度不同的两个人得到的频谱有明显区别，而这一区别可通过将频谱在频率上进行压缩或拉伸来模拟。因此，如果我们设定一个标准声道长度，则其它声道长度的频谱即可通过一个形变因子α归整到该标准频谱上来。这一技术称为声道长度规整（VTLN），形式化如下：

$$S^{\alpha}(\omega) = S(\alpha\omega)$$

其中$S(\omega)$为该发音人的原始频谱，$S^{\alpha}(\omega)$为归整后的频谱。在实际系统中，一般采用分段线性映射函数来实现非线性规整，如图1所示。

在VTLN需要估计每个说话人的形变因子 α 。通常的作法是基于一段该说话人的语音，尝试不同的 α 取值，找到一个最优取值使得依该值对语音进行归整后在参考模型下概率最大化。由于形变因子是应用在频域上的，对以MFCC为特征的系统来说，需要多次生成特征。线性VTLN可以在特征域上对不同形变因子设计线性映射，可免除重复生成特征的麻烦[3]。

VTLN具有明确的物理意义，实现简单，在语音识别中得到广泛应用。然而，一些研究也发现VTLN事实上可以通过特征上线性变换进行补偿，因此VTLN 在一些实际系统中的作用可能并不明显[4]。关于特征上的线性变换，我们将在下节介绍。

Kaldi中包含了VTLN的计算方法，如图9所示。同时，Kaldi wsj recipe中也提供了VTLN可选操作（缺省是关闭的），如图10所示。

3 声学模型自适应：HMM-GMM系统

在HMM时代，典型的声学模型是HMM-GMM架构，如图4所示，其中HMM (隐马尔可夫模型)用来描述信号动态特性，GMM（高斯混合模型）用来描

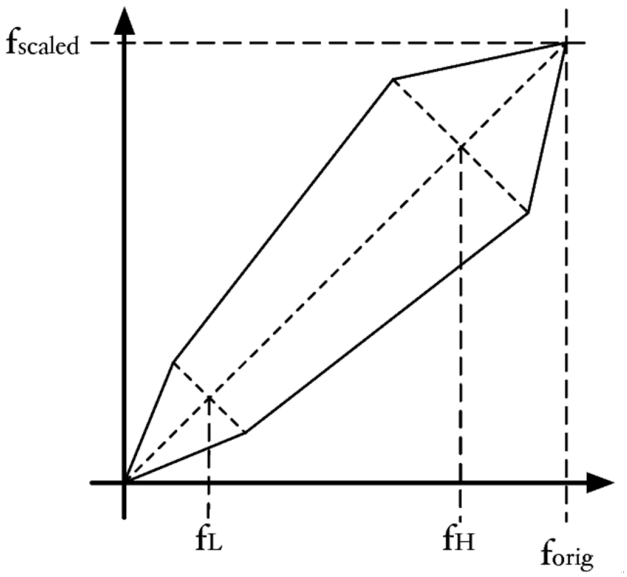

Figure 1: 分段线性VTLN函数。横轴为原始频率，纵轴为变换后的频率。中间虚线代表参考声道长度(α=1)下的映射，上下两条实线分别代表不同形变因子对应的映射函数。。

```
BaseFloat MelBanks::VtlnWarpFreq(BaseFloat vtln_low_cutoff,  // upper+lower frequency cutoffs for VTLN.
                                 BaseFloat vtln_high_cutoff,
                                 BaseFloat low_freq,  // upper+lower frequency cutoffs in mel computation
                                 BaseFloat high_freq,
                                 BaseFloat vtln_warp_factor,
                                 BaseFloat freq) {
  /// This computes a VTLN warping function that is not the same as HTK's one,
  /// but has similar inputs (this function has the advantage of never producing
  /// empty bins).

  /// This function computes a warp function F(freq), defined between low_freq and
  /// high_freq inclusive, with the following properties:
  ///  F(low_freq) == low_freq
  ///  F(high_freq) == high_freq
  /// The function is continuous and piecewise linear with two inflection
  ///   points.
  /// The lower inflection point (measured in terms of the unwarped
  ///  frequency) is at frequency l, determined as described below.
  /// The higher inflection point is at a frequency h, determined as
  ///   described below.
  /// If l <= f <= h, then F(f) = f/vtln_warp_factor.
  /// If the higher inflection point (measured in terms of the unwarped
  ///   frequency) is at h, then max(h, F(h)) == vtln_high_cutoff.
  ///   Since (by the last point) F(h) == h/vtln_warp_factor, then
  ///   max(h, h/vtln_warp_factor) == vtln_high_cutoff, so
  ///   h = vtln_high_cutoff / max(1, 1/vtln_warp_factor).
  ///     = vtln_high_cutoff * min(1, vtln_warp_factor).
  /// If the lower inflection point (measured in terms of the unwarped
  ///   frequency) is at l, then min(l, F(l)) == vtln_low_cutoff
  ///   This implies that l = vtln_low_cutoff / min(1, 1/vtln_warp_factor)
  ///                       = vtln_low_cutoff * max(1, vtln_warp_factor)
```

Figure 2: Kaldi中src/feat/mel-computations.cc中实现的VTLN代码。

```
    # Demonstrating Minimum Bayes Risk decoding (like Confusion Network decoding):
    mkdir exp/tri2b/decode_nosp_tgpr_${data}_tg_mbr
    cp exp/tri2b/decode_nosp_tgpr_${data}_tg/lat.*.gz \
       exp/tri2b/decode_nosp_tgpr_${data}_tg_mbr;
    local/score_mbr.sh --cmd "$decode_cmd"  \
       data/test_${data}/ data/lang_nosp_test_tgpr/ \
       exp/tri2b/decode_nosp_tgpr_${data}_tg_mbr
  done
fi

# At this point, you could run the example scripts that show how VTLN works.
# We haven't included this in the default recipes.
# local/run_vtln.sh --lang-suffix "_nosp"
# local/run_vtln2.sh --lang-suffix "_nosp"
fi
```

Figure 3: Kaldi中wsj recipe下的VTLN步骤。

述HMM每个状态的准静态特性。HMM-GMM 模型的一个特点是结构简单，参数的物理意义直观明了。因此，只需要对那些与说话人特性相关的参数进行适当调整，即可实现对模型的快速自适应。

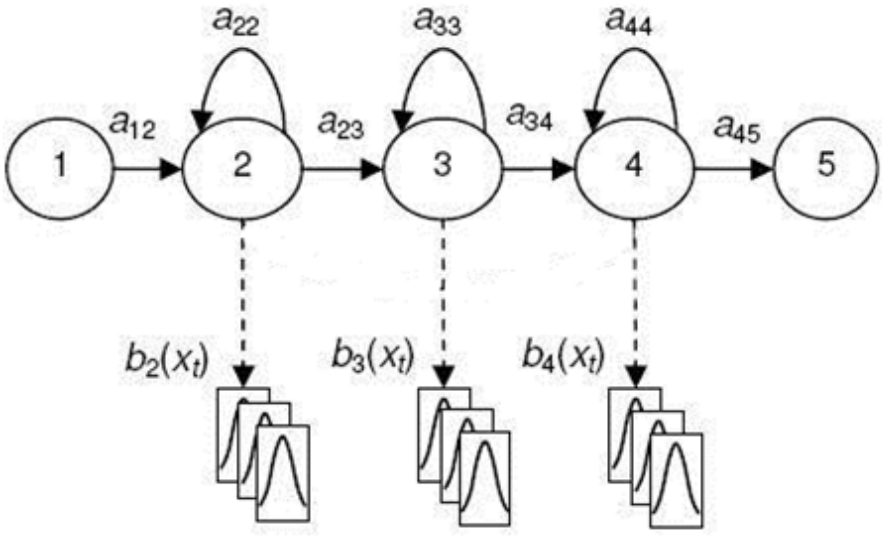

Figure 4: HMM-GMM模型，其中HMM包括三个输出状态，每个状态用一个GMM表示。

研究表明，HMM模型对说话人特性的表征并不明显，因此绝大多数自适应方法是对GMM模型的调整。一个GMM模型包含若干高斯成分，每个高斯成分是一个高斯分布，其参数包括一个均值向量，一个协方差矩阵，以及一个在GMM中的权重比例。说话人自适应的任务是通过调整均值、协方差、权重这

三个参数，使得调整后的GMM对目标说话人有更好的描述。更精确地说，是使得目标说话人的自适应数据在调整后的GMM中概率最大化。

常用的参数调整方法有两种：最大后验概率估计（Maximum a Posterior, MAP）[5]和最大似然线性回归（Maximum Likelihood Linear Regression）[6, 7]。

3.1 基于MAP的自适应方法

我们知道，一个语音识别系统中包含多个音素（Phone），每个音素基于决策树（Decision Tree）扩展为若干上下文相关音素（CD phone）。在HMM系统中，每个上下文相关音素由一个HMM建模。如果可以将自适应语音中的每个语音帧合理地分配到所属的HMM模型、HMM状态以及该状态的某一个高斯成份，我们将可以基于每个高斯成份所对应的语音帧对该高斯成份的参数进行调整。基于同样的准则，实验表明，在均值、协方差、权重这三类参数中，对均值的调整效果最为明显，因此，我们将只考虑对均值的更新。

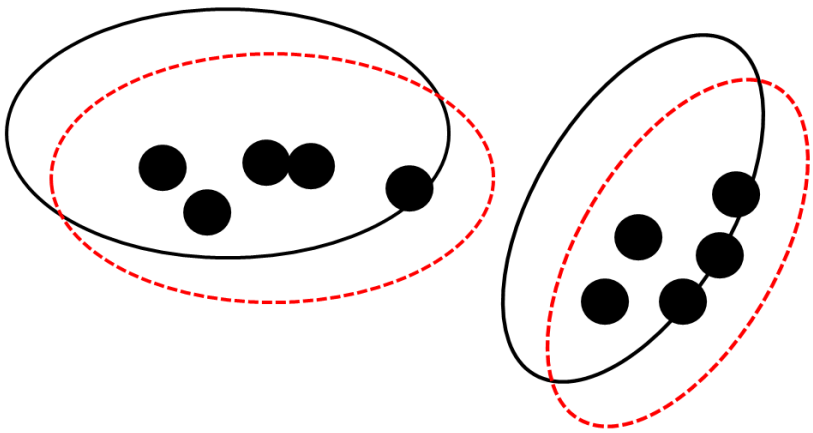

Figure 5: GMM模型的自适应。黑圈表示原始GMM的两个高斯成分，实心黑点表示目标说话人的语音特征向量。可见，这两个高斯成分无法有效描述目标说话人语音。通过对模型进行自适应，两个高斯成分发生了变化，如图中红色虚线所示。

不失普遍性，设待更新的高斯成分为N_{ijk}，其中i表示HMM序号，j表示状态序号，k表示高斯成分序号；对应该高斯成分的语音帧为X_{ijk} =

$\{x_{ijk}(1), x_{ijk}(2), , , x_{ijk}(T_{ijk})\}$。依最大似然（Maximum Likelihood）准则，可计算出该高斯成分的均值如下：

$$\mu_{ijk} = \frac{1}{T_{ijk}} \sum_{t=1}^{T_{ijk}} x_{ijk}(t). \tag{1}$$

然而，如果该高斯成分分配到的语音帧较少，上述ML估计显然无法得到合理的均值。为此，我们可以将上述ML估计得到的均值与原始通用模型的均值做一个线性插值，得到自适应后的均值如下：

$$\mu'_{ijk} = (1-\alpha)\mu^0_{ijk} + \alpha\mu_{ijk}. \tag{2}$$

其中μ^0_{ijk}为原始通用模型中对应的高斯成分的均值。可以证明，上述均值估计方式可以通过一个最大后验概率估计得到，其中先验概率是一个以μ^0_{ijk}为均值，以$\hat{\Sigma}_{ijk}$为协方差距阵的高斯分布。通常假设$\hat{\Sigma}_{ijk}$为对角的，且有：$\hat{\Sigma}_{ijk} = \hat{\sigma}I$。该先验概率写成公式如下：

$$p(\mu_{ijk}) = N(\mu_{ijk}|\mu^0_{ijk}, \sigma_{\hat{ijk}}I)$$

对应的条件概率是对自适应语音数据的高斯分布，同样取对角协方差，有：

$$p(X_{ijk}|\mu_{ijk}) = \prod_t N(x_{ijk}(t)|\mu_{ijk}, \sigma_{ijk}I)$$

则依贝叶斯公式，有：

$$p(\mu_{ijk}|X_{ijk}) \propto p(\mu_{ijk})p(X_{ijk}|\mu_{ijk})$$

上述后验概率取最大值的μ_{ijk}具有式（2）所示的形式，其中：

$$\alpha = \frac{T_{ijk}\hat{\sigma}_{ijk}}{T_{ijk}\hat{\sigma}_{ijk} + \sigma_{ijk}}$$

仔细观察式上式可以发现，当自适应数据较少时，T_{ijk} 较小，α趋近于零，则式2中的μ'_{ijk}趋近于原始模型的均值μ^0_{ijk}，即模型没有更新。换句话说，当没有多少自适应数据时，尽量采用原模型。反之，如果自适应数据较多，T_{ijk}较大，α趋近于1，则μ'_{ijk}趋近于ML估计μ_{ijk}。这意味着数据已经足够充分，用自适应数据得到的估计已经可以完全信任了。这一性质使得MAP非常灵活，可以依实际训练数据的多少选择合理的自适应模型。

值得说明的是，上述自适应方法假设每个语音帧被分配到合理的音素、状态和高斯成分上，这事实上是不可能的。尽管如此，我们依然可以利用一些对

齐算法对语音帧进行“软性”分配，即将语音帧依概率分配到不同的高斯成分上；在进行模型更新时，只需依这一分配概率对不同语音帧进行加权处理即可。这一对齐过程即是我们在Kaldi 的recipe中经常看到的alignment操作。下图给出wsj recipe中在训练tri1之前的alignment过程。注意该过程一般只能分配到音素或状态，对高斯成分的软性分配可以通过求各高斯成分的后验概率实现。

```
if [ $stage -le 2 ]; then
  # tri1
  if $train; then
    steps/align_si.sh --boost-silence 1.25 --nj 10 --cmd "$train_cmd" \
      data/train_si84_half data/lang_nosp exp/mono0a exp/mono0a_ali || exit 1;

    steps/train_deltas.sh --boost-silence 1.25 --cmd "$train_cmd" 2000 10000 \
      data/train_si84_half data/lang_nosp exp/mono0a_ali exp/tri1 || exit 1;
  fi

  if $decode; then
    utils/mkgraph.sh data/lang_nosp_test_tgpr \
      exp/tri1 exp/tri1/graph_nosp_tgpr || exit 1;

    for data in dev93 eval92; do
      nspk=$(wc -l <data/test_${data}/spk2utt)
      steps/decode.sh --nj $nspk --cmd "$decode_cmd" exp/tri1/graph_nosp_tgpr \
        data/test_${data} exp/tri1/decode_nosp_tgpr_${data} || exit 1;
```

Figure 6: Kaldi wsj recipe中提供的alignment过程。

3.2 基于MLLR的自适应方法

在MAP自适应方法中，对每个高斯成分的自适应是独立进行的。这一独立更新方案使得MAP非常灵活；另一方面，由于不同音素、状态、高斯成分之间无法共享数据，使得那些没有分配到自适应数据的高斯成分无法更新。一种解决方法是设计一个对所有高斯成分进行统一更新的变换M，使得任何一个高斯成分上分配到的自适应数据都可以对全体高斯成分产生影响。如果M是一个线性变换，则经过该变换得到均值向量为:

$$\mu'_{ijk} = M\mu_{ijk}.$$

选择M使得变换后的模型对自适应数据的生成概率最大化。这一优化准则写成似然函数形式如下：

$$L(M) = \prod_{ijk} p(x_{ijk}|M\mu_{ijk}, \sigma_{ijk}). \tag{3}$$

注意上述似然函数包括所有音素、状态、高斯成分。我们的基本思路是估计一个线性变换，使得式（3）所示的似然函数最大化，因此称为最大线性似然回归，即MLLR。注意，上式中我们依然假设对语音帧进行了较好的分配。在实际系统中，只能做到“软性”分配，需要利用alignment 算法。值得强调的是，

MLLR算法中的变换矩阵M是“全局”的，为所有高斯成分共享，这意味着某些高斯成分即使没有分配到自适应数据，依然可以得到更新，这是和MAP方法的显著区别。这一方面意味着MLLR 自适应需要较少的数据量即可实现自适应，另一方面，这一特性也意味着各个高斯成分无法实现”渐近最优化”，即当数据量足够充分时，MLLR并不能逼近ML估计，因此性能存在上限。相对而言，数据足够多时，MAP是可以接近ML估计的。

MLLR算法仅对均值进行更新。如果我们希望同时对方差进行变换，计算上将比较复杂。然而，存在一种特殊情况：当对方差的变换和对均值变换相匹配时，计算将非常简单。具体而言，我们希望均值的更新具有如式（3）所示的形式，对均值的更新具有如下对应形式：

$$\Sigma' = M\Sigma M^T.$$

在这一条件下，语音帧的概率密度函数可整理如下：

$$p(X_{ijk}|M\mu_{ijk}, M\Sigma M^T) = p(MX_{ijk}|\mu_{ijk}, \Sigma).$$

这意味着对均值和方差的匹配更新等价于保持原始模型不变，仅对语音特征向量做如下线性变换：

$$x'_{ijk} = Mx_ijk”$$

这种在特征上进行线性变换的方法称为fMLLR。

3.2.1 SAT训练

fMLLR可以用来训练通用说话人模型。这一方法假设一个人的发音可以通过一个fMLLR从说话人特征中去掉说话人相关信息，再利用该“中性”特征训练一个通用说话人模型。基于该中性模型，可再次更新每个人的fMLLR，得到新的中性特征。上述过程可迭代进行（如图7所示），称为SAT训练（Speaker Adaptive Training，SAT)[8]。中性模型不包含说话人信息，因此可实现更好的建模。在实际进行识别时，首先需要基于中性模型计算个人的fMLLR，再将该fMLLR应用到待识别语音的特征向量，并基于中性模型进行识别。

Kaldi recipe中提供了SAT训练的脚本，如图所示。

4 声学模型自适应：DNN系统

现代语音识别系统基于深度神经网络（DNN)。依动态模型的不同，当前主流框架包括DNN-RNN系统和DNN-HMM 系统。为描述方便，我们只讨论DNN-HMM系统，但相应方法可同样应用于DNN-RNN系统。

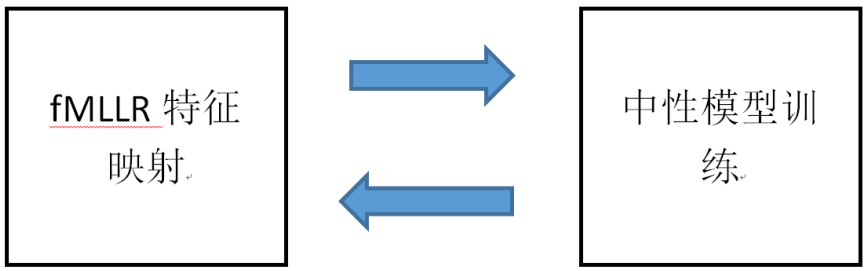

Figure 7: SAT训练。基于fMLLR的特征映射和基于映射后特征的模型训练迭代进行。

```
# local/run_delas.sh trains a delta+delta-delta system.  It's not really recommended or
# necessary, but it does contain a demonstration of the decode_fromlats.sh
# script which isn't used elsewhere.
# local/run_deltas.sh

if [ $stage -le 4 ]; then
  # From 2b system, train 3b which is LDA + MLLT + SAT.

  # Align tri2b system with all the si284 data.
  if $train; then
    steps/align_si.sh  --nj 10 --cmd "$train_cmd" \
      data/train_si284 data/lang_nosp exp/tri2b exp/tri2b_ali_si284  || exit 1;

    steps/train_sat.sh --cmd "$train_cmd" 4200 40000 \
      data/train_si284 data/lang_nosp exp/tri2b_ali_si284 exp/tri3b || exit 1;
  fi
```

Figure 8: Kaldi wsj recipe中提供的SAT训练脚本。

在DNN-HMM系统中，动态模型是一个HMM，其中每个状态的输出概率是基于DNN所生成的后验概率计算得到的。和GMM不同，DNN的参数数量庞大且相互依赖，对某些参数的修改可能会对结果产生显著影响。因此如果想通过更新模型来实现说话人自适应，需要特别设计可更新的参数集以及更新算法。除了对DNN模型进行直接更新，另一种主流方法是是基于说话人特征的条件学习。我们将分别介绍这两种方法。

4.1 模型参数自适应学习

如前所述，DNN的参数数量庞大且相互依赖，在仅有少量自适应数据的前提下，对所有参数进行更新存在过拟合的风险。过拟合后，模型将在当前的自适应数据性能有显著提高，但对来自该说话人的其它数据性能无法提高甚至下降。

为解决这一问题，通常选择DNN模型中的某一层进行更新，从而保证模型不会偏离原始模型太远。研究者尝试了更新输入层、隐藏层和输出层等各种方

案，发现更新隐藏层效果更好[9]。另一种方式是在DNN中加入一个自适应层，将其初始化为对角矩阵，从而保持整个DNN的映射函数不变。在自适应时，仅更新该自适应层，甚至仅更新该层矩阵的对角值，从而降低过拟合的风险。还有研究者对某一层矩阵进行SVD分解，自适应时仅对分解后的特征向量进行更新[10]。最后，有研究者对输入的特征向量进行线性变换，在不改变DNN模型的前提下，使变换后的特征性能更好[11]。这一方法事实上等价于在DNN的输入端加入一个自适应层并对该层进行更新。

微软的研究者提出一种基于相对熵约束的自适应训练方法，该方法在进行自适应时，优化目标不仅是模型在自适应语料上性能更好，同时希望新模型的输出和原模型的输出不要相差太远[12]。在语音识别中，DNN的输出为不同pdf ID上的后验概率，因此输出之间的差异性可用相对熵来衡量，即更新后的输出与原始输出之间的相对熵不宜过大。基于该约束，可以对网络中的所有参数进行训练。

上述参数自适应学习方法可对模型进行有限更新，但是在操作时需要仔细考虑平衡自适应数据量与学习率之间的关系，通常不易处理。

4.2 基于说话人向量的条件学习

另一种主要的DNN说话人自适应方法是基于说话人向量的条件学习，如图9 所示。在该模型中，DNN的输入不仅包括传统声学特征（如Fbank ），还包括一个表征说话人特性的说话人特征（向量）。说话人特征可以通过概率统计或深度学习方法生成，其中基于概率模型的i-vector方法最为常用[13]。在该方法中，基于一个混合线性高斯模型，将语音特征序列中的说话人特性（称为i-vector）抽取出来，作为DNN的辅助输入。识别时，只需用同样的模型将待识别说话人的i-vector提取出来，即可实现说话人自适应。这一方法只需极少量语音（几帧）即可实现对说话人特征的提取，简洁高效。更有意义的是，因为i-vector是对一段语音信号的整体描述，因此这一方法不仅可以用于说话人自适应，还可以实现不同环境下的模型自适应。关于说话人识别和i-vector的更多相关知识，将在后续章节详细讨论。

Kaldi的recipe中提供了基于i-vector的条件学习脚本，如图10所示。

5 领域自适应

我们已经大略介绍了对单一说话人的自适应方法。事实上，类似的方法也可以用于特定人群的自适应上，例如对某一口音的自适应（如河南口音、东北口音等）或对某一应用境的自适应（如公交、大街、咖啡店等）。这种面向某一

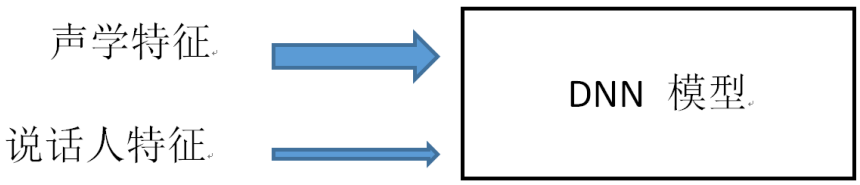

Figure 9: 基于i-vector的DNN条件学习方法。

```
if ! cuda-compiled; then
  cat <<EOF && exit 1
This script is intended to be used with GPUs but you have not compiled Kaldi with CUDA
If you want to use GPUs (and have them), go to src/, and configure and make on a machine
where "nvcc" is installed.
EOF
fi

local/nnet3/run_ivector_common.sh --stage $stage --nj $nj \
                                  --train-set $train_set --gmm $gmm \
                                  --num-threads-ubm $num_threads_ubm \
                                  --nnet3-affix "$nnet3_affix"



gmm_dir=exp/${gmm}
ali_dir=exp/${gmm}_ali_${train_set}_sp
dir=exp/nnet3${nnet3_affix}/tdnn${tdnn_affix}_sp
train_data_dir=data/${train_set}_sp_hires
train_ivector_dir=exp/nnet3${nnet3_affix}/ivectors_${train_set}_sp_hires

for f in $train_data_dir/feats.scp $train_ivector_dir/ivector_online.scp \
    $gmm_dir/{graph_tgpr,graph_bd_tgpr}/HCLG.fst \
    $ali_dir/ali.1.gz $gmm_dir/final.mdl; do
  [ ! -f $f ] && echo "$0: expected file $f to exist" && exit 1
done
```

Figure 10: Kaldi wsj recipe中，nnet3/run_tdnn.sh中基于i-vector的条件学习代码。

特定场景的自适应可称为领域自适应。和说话人自适应相比，领域自适应一般数据量更大，对模型的更新力度更大。另外，在领域自适应中，除了对声学特性的修正，在语言模型上也需要进行相应的调整，特别是对领域专有名词的处理以及相领域语言模型的训练。对n-gram语言模型来说，一般采用新旧模型插值法实现语言模型自适应[14]。对神经网络语言模型的自适应方法研究的还比较少[15]。

6 小结

本节介绍了说话人自适应的若干方法。总体来说，这些方法可以分为特征域自适应和模型域自适应。特征域自适应保持模型不变，对特征进行说话人相关的变换，以增加与模型的匹配度，如fMLLR和VTLN；模型域方法通过更新模型参数实现对特定说话人语音特性的更好描述。总体来说，特征域变换更灵活，需要的数据较少；模型域方法需要更多数据，但通常对说话人特性的学习更细致，性能也更好。

在模型方法中，一般对状态生成模型进行自适应，即GMM模型或DNN模型。相对而言，基于其清晰的概率结构，GMM模型的自适应更加有效; DNN模型的参数高度共享，互相依赖，修改不易。目前的主流方法是在DNN模型的输入端加入一个表征说话人特性的辅助特征，将说话人特性纳入到模型之中，通常可取得较好的效果。

References

[1] D. Wang and T. F. Zheng, "Transfer learning for speech and language processing," *arXiv preprint arXiv:1511.06066*, 2015.

[2] L. Lee and R. Rose, "A frequency warping approach to speaker normalization," *IEEE Transactions on speech and audio processing*, vol. 6, no. 1, pp. 49–60, 1998.

[3] D. R. Sanand, D. D. Kumar, and S. Umesh, "Linear transformation approach to vtln using dynamic frequency warping," in *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[4] X. Cui and A. Alwan, "Mllr-like speaker adaptation based on linearization of vtln with mfcc features," in *Ninth European Conference on Speech Communication and Technology*, 2005.

[5] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *IEEE transactions on speech and audio processing*, vol. 2, no. 2, pp. 291–298, 1994.

[6] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models," *Computer speech & language*, vol. 9, no. 2, pp. 171–185, 1995.

[7] M. J. Gales *et al.*, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[8] T. Anastasakos, J. McDonough, and J. Makhoul, "Speaker adaptive training: A maximum likelihood approach to speaker normalization," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, vol. 2. IEEE, 1997, pp. 1043–1046.

[9] H. Liao, "Speaker adaptation of context dependent deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7947–7951.

[10] J. Xue, J. Li, D. Yu, M. Seltzer, and Y. Gong, "Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6359–6363.

[11] S. Xue, H. Jiang, L. Dai, and Q. Liu, "Speaker adaptation of hybrid nn/hmm model for speech recognition based on singular value decomposition," *Journal of Signal Processing Systems*, vol. 82, no. 2, pp. 175–185, 2016.

[12] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7893–7897.

[13] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors." in *ASRU*, 2013, pp. 55–59.

[14] A. Stolcke, "Srilm-an extensible language modeling toolkit," in *Seventh international conference on spoken language processing*, 2002.

[15] M. Ma, M. Nirschl, F. Biadsy, and S. Kumar, "Approaches for neural-network language model adaptation," *Proc. Interspeech, Stockholm, Sweden*, pp. 259–263, 2017.