

语音识别基本法

清华大学语音和语言技术中心

© 清华大学语音和语言技术中心

<http://cslt.org>

本书不是书

有一天，王叔说，我们要写一本书，于是便有了这本“书”。

物有本末，事有终始。一件事情，从头到尾反复地经历与琢磨下来，心中的大图越来越细致完整，慢慢地形成了一套自己的行事法则，这就是经验了。任何一个新的领域，都有经验可言。

我们从实习生开始，摸索着起来，趟过了很多语音领域的坑坑洼洼。后来，与新的实习生一起工作学习，发现有些经验是固定不变的，常限于口口相传，费时耗力又难成体系，不如写在纸上。于是，我们决定把一些经验记录下来，与后来的小伙伴们交流互进，并不断充实完善，也起到一点知识传承的作用。

经验的主观性使得本书并不如论文一样义正辞严，不连贯性也使得本书不能像教材一样循循善诱。本书是直觉的首先流露，多重于实践上的“是什么”，让新来者尽快“感受”到整个语音识别系统的大体模样，做到心中有谱，会使用 Kaldi 等工具调配常见的系统结构，理论上的“为什么”待以后点点积淀。内外兼修，方能见长，实验室王叔（wangd.cs.lt.org）写的《机器学习》恰可作为理论上的支撑。

本书涉及的代码放在 <https://github.com/tzyll/kaldi> 中，该分支定期更新至最新的 Kaldi 状态。本书牵扯的事，乃至无关的事，都可往来讨论，邮件可至 tangzy@cs.lt.org。尽信书不如无书，行色匆匆，以此免责：)

此外

风声雨声读书声，声声入耳。我们关心所有的声音，并渴望用手中的技术发掘其中的意义，使之广为流传，从而创造更多的价值，影响更多的人。

进入这个实验室时，我们与语音的缘分便注定了。我们的生活和工作与语音技术分不开了，我们的理想和抱负也惯用了语音领域的思维，这是我们受益的地方，更是技术分子的归属感：更广阔的声音、更深远的智慧，始于语音，始于初次听见。

实验室期待拥有同一个目标、能够一起做事的人。博士后、工程师、联合培养学生和实习生，构成了实验室的生物多样性，更是语音领域的源头活水。

目录

I

语音识别基础

1	语音是什么	9
2	语音识别方法	13
3	语音识别工具	19

II

语音识别基本流程

4	实验先行	23
5	前端处理	31
6	训练与解码	39

III

语音识别实际问题

7	说话人自适应	47
8	噪声对抗与环境鲁棒性	57

9	新词处理与领域泛化	69
10	小语种识别	71
11	关键词唤醒与嵌入式系统	77

IV

前沿课题

12	说话人识别	85
13	语种识别	87
14	情绪识别	89
15	语音合成	91

V

Roman Forum

16	技术收藏夹	105
17	Q & A	107
	参考文献	109
	索引	115



语音识别基础

1	语音是什么	9
1.1	大音希声	
1.2	看见语音	
2	语音识别方法	13
2.1	总体思路	
2.2	实现方法	
3	语音识别工具	19
3.1	Kaldi	
3.2	深度学习平台	

1. 语音是什么

by 阿汤

从最起初的一声巨响，到梵音天籁，到耳旁的窃窃私语，到妈妈喊我回家吃饭，总离不开声音。声音是这个世界存在并运动着的证据。

1.1 大音希声

假设我们已经知道了声音是什么。

我们可以找到很多描述声音的词语，如“抑扬顿挫”、“余音绕梁”。当我们在脑海中搜刮这类词语时，描述对象总绕不过这两个：人的声音和物的声音。人的声音，就是语音；物的声音，多数想到的是音乐。这样的选择源于人的先验预期：语音和音乐才最可能有意义，有意义的才去关注。估计不会有人乐于用丰富的辞藻来描述毫无意义的声音。所以，语音研究的意义在于语音本身所传递的意义是什么，以及语音为什么能够传递意义。

声音有很多，每时每刻每次振动都能产生声音，可是有意义的声音实在不多。我们可以使用机器随机生成一段声音，心想着也许这段声音可以产生一些文字内涵。这个想法与很多年前就开始忙不迭地敲打莎士比亚巨著的大猩猩没有差别。不管重复多少次，这些随机的声音听起来都是杂音，没意思。很显然，在这样一个庞大的声音空间中，有意义的语音和音乐只是其中极微小的一点，这也是“大音希声”的一种解释吧。偏偏人类就能毫不费力找到那个点，并且能说会道，这种搜索能力也是千百年来才积攒下来的。不过就算是这么一个小点，古往今来的文学和音乐经典也并未占据多少地盘，这也使得语音语言的研究、文学音乐的创作有着广阔的空间。

从大音希声中，我们可以得到以下一些启示：语言是高度概括和规范化的产物，它的熵值（简单理解为系统的混乱程度）极低，所以语言本身反映了一种思维方式，比如不同

语言对“过去时”、“现在时”、“将来时”的处理方式体现了对时间的不同感受，不同语言对主谓宾的排序体现了对空间层次的不同感知；还有，语音在声音空间中是高度集中的，这使得我们在解析一段语音时不用搜索整个声音空间，少了一些盲目性（不过语言本身的博大精深已算是无垠了）。

1.2 看见语音

语音是用来听的，看不见，摸不着，但是我们可以看看语音的保存形式。自然存在的语音是连续的波动，具有波的所有属性，因而声波可以保存成离散的数字，即模数转换（Analog to Digital Conversion, ADC），所以，我们之后所研究的语音并不是声音的最原始形态，甚至都不叫声音，一串数字而已，但这些数字却达到了它的目的：再现声音，且需要传递的信息不丢失。音乐可以做得更彻底，直接将声音记录在一纸没有动静的乐谱上。除了声音，光线也是自然存在的现象，同样地，我们也可以将它数字化，保存成图片或视频。机器学习中注重表征学习（Representation Learning），不管是声音还是光影，它们的数字化保存形式已经是一种表征方法了。对文本的处理显得直来直去一些，因为文字是人类发明出来的，发明文字的目的就是为了保存，如音符一样，它也是一种离散的可记录、传播的符号，它的形态就是它的保存形式，所以文字本身就是文本处理的原始表征方法。



图 1.1: 语音文件的波形图（Adobe Audition 生成）

语音的基本保存形式可用波形图（Waveform）展现出来，如图 1.1 所示，可以简单地看作是一串上下摆动的数字，比如，每 1 秒的音频可以用 16000 个电压数值表示，即采样率为 16 kHz 。进一步聚焦放大波形图，可以清晰地看到每个采样点，如图 1.2 所示。真正的语音不需要额外的注解，但对于数字化的语音来说，还需要额外的信息对文件格式进行说明，如信道、采样率、精度、时长等，并有文件大小 = 格式信息 + 信道数*采样率*精度*时长。可以用 `soxi` 查看文件信息，如图 1.3 所示。

语音，是包含时序信息的序列，是时域表征。离散傅里叶变换（Discrete Fourier Transform, DFT）使得语音的频域分析成为可能，图 1.1 的语音可以变成图 1.4 的频谱图（Spectrogram）模样，图中可以清楚地看到“层峦叠嶂”，原始音频里的信息又以另一种表征方法释放出来了，颜色明暗表示数值大小，较亮的条纹即是共振峰（Formant）。整



图 1.2: 语音文件的采样点 (Adobe Audition 生成)

```
[tangzy@csilt.org ~]$ soxi 1a_1.wav
Input File      : '1a_1.wav'
Channels        : 1
Sample Rate     : 16000
Precision       : 16-bit
Duration        : 00:00:03.54 = 56703 samples ~ 265.795 CDDA sectors
File Size       : 119k
Bit Rate        : 268k
Sample Encoding: 16-bit Signed Integer PCM
```

图 1.3: 语音文件的格式信息 (Linux 系统)

个过程就好比一双好耳朵听到了一首随时间流动的曲子，随即写出了它的谱子，看着谱，曲子又随即可以复现出来。时域与频域之间是一一对应的，可以代表彼此。从一种表征到另一种表征，包含的意义都在，只是有些藏得深，挖掘不到，有些露得浅，一目了然，后者才是适合机器的，所以机器学习领域常常脱不开表征学习的本质。

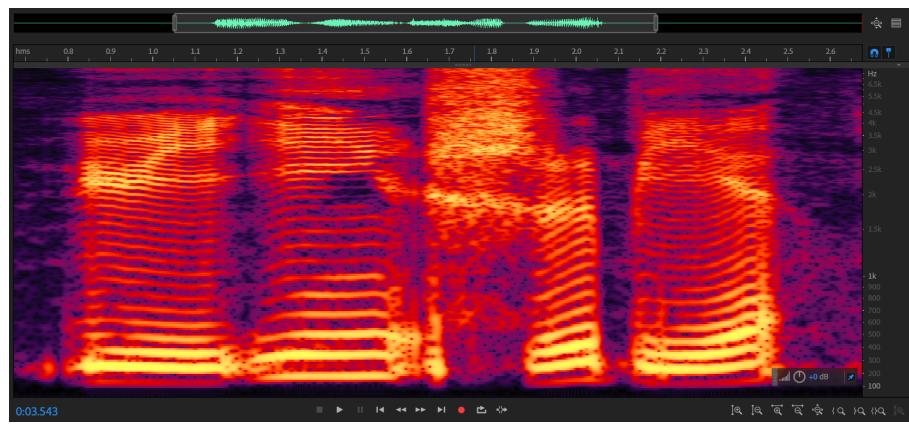


图 1.4: 语音文件的频谱图 (Adobe Audition 生成)

2. 语音识别方法

by 阿汤

语音识别的全称是自动语音识别（Automatic Speech Recognition, ASR），说得多了，就把“自动”省去了，认为“自动”是理所当然的了。语音识别属于序列转换技术，它将语音序列转换为文本序列。大体来说，这是一次搬运，是把一段话的表现形式从语音变成了文本，至于文本想要表达的深层含义（自然语言理解）、倾诉的感情（情感识别）、说话的主体（说话人识别），就需要其他的系统来处理，所以语音应用开始时是分工明确的，但这显然不符合人类的感知和认知，所以后来的技术也有了不同程度的整合和联合学习。

如何实现有效的语音识别，无非是，先确定问题，然后找个模型，最后训好它。

2.1 总体思路

已知一段语音信号，处理成声学特征向量（Feature Vector，而不是 Eigenvector）后表示为 $X = [x_1, x_2, x_3, \dots]$ ，其中 x_i 表示一帧（Frame）特征向量，可能的文本序列表示为 $W = [w_1, w_2, w_3, \dots]$ ，其中 w_i 表示一个词，求 $W^* = \operatorname{argmax}_W P(W|X)$ ，这是语音识别的基本出发点。可知

$$P(W|X) = \frac{P(X|W)P(W)}{P(X)} \quad (2.1)$$

$$\propto P(X|W)P(W) \quad (2.2)$$

其中， $P(X|W)$ 称之为声学模型（Acoustic Model）， $P(W)$ 称之为语言模型（Language Model），二者对语音语言现象刻画得越深刻，识别结果越准确。化整为零，逐个击破，很符合逻辑惯性，所以大多数研究都把语音识别分作声学模型和语言模型两部分，并把很多精力放在声学模型的改进上。后来，基于深度学习和大数据的端对端（End-to-End）方法发展起来，它直接计算 $P(W|X)$ ，把声学模型和语言模型融为一体。

T 对于不同的候选文本来说，待解码语音的概率保持不变，是各文本之间的不变量，所以公式 2.1 中的 $P(X)$ 可以省去不算。

2.2 实现方法

一段语音，经历什么才能变成它所对应的文本呢？语音作为输入，文本作为输出，第一反应是该有一个函数，自变量一代入，结果就出来了。可是，由于各种因素（如环境、说话人等）的影响，同一段文本，读一千遍就有一千个模样，语音的数字化存储也因之而不同，长短不一，幅度不一，就是一大堆数字的组合爆炸，想要找到一个万全的规则将这些语音唯一地对应到同一段文本，这是演算逻辑所为难的；而常用的词汇量也很庞大，能够拼成的语句不计其数，面对一段语音，遍历搜寻所有可能的文本序列必然无法负担。这样，定义域和值域都是汪洋大海，难以通过一个函数一步到位地映射起来。

如果我们能够找到语音和文本的基本组成单位，并且这些单位是精确的、规整的、可控的，那么二者之间的映射关系会单纯一些。语音，选择的基本单位是帧（Frame），帧是一个向量，整条语音可以整理为以帧为单位的向量组，每帧维度固定不变。一帧数据是由一小段语音经由 ASR 前端数字信号处理产生，涉及的主要技术包括离散傅里叶变换和梅尔滤波器组（Mel Filter Bank）等。一帧的跨度是可调的，以适应不同的文本单位。对于文本，字（或字母）组成词，词组成句子，字词是首先想到的组成单位。

至此，语音的基本组成单位有了统一的格式，文本的基本组成单位又是有限的，问题便在于如何将二者对应起来，如图 2.1 归纳了当下常用的路数，差异多体现在声学模型上，大体说来有以下几个关键结构。

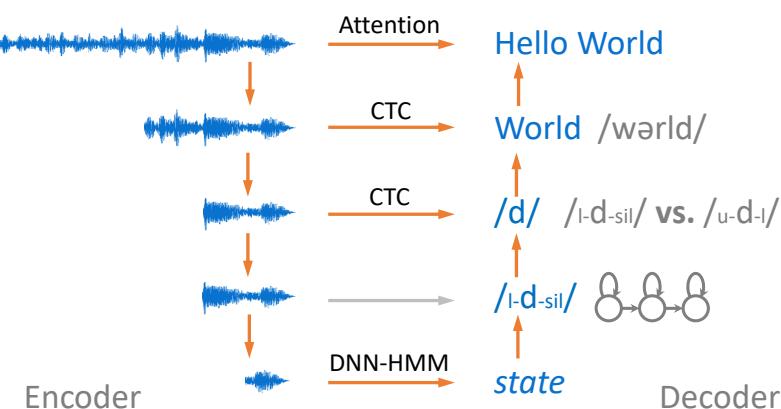


图 2.1: 语音识别的基本途径

2.2.1 HMM

一个首要问题是，语音和文本的不定长关系，使得二者的序列之间无法一一对应，常规的概率公式演算就不适宜了。隐马尔可夫模型（Hidden Markov Model, HMM）正好可以解决这个问题。比如 $P(X|W) = P(x_1, x_2, x_3|w_1, w_2)$ 可以表示成如图 2.2 隐马尔科夫链的

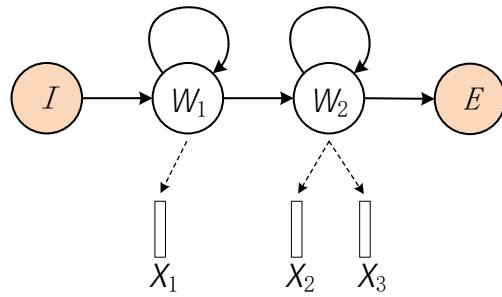


图 2.2: 隐马尔科夫模型

形式，图中 w 是 HMM 的隐含状态， x 是 HMM 的观测值，隐含状态数与观测值数目不受彼此约束，这便解决了输入输出的不定长问题，并有

$$P(X|W) = P(w_1)P(x_1|w_1)P(w_2|w_1)P(x_2|w_2)P(w_3|w_2)P(x_3|w_2) \quad (2.3)$$

其中，初始状态概率 ($P(w_1)$) 和状态转移概率 ($P(w_2|w_1)$ 、 $P(w_3|w_2)$) 可以用常规的统计方法从样本中计算出来，主要的难点在于发射概率 ($P(x_1|w_1)$ 、 $P(x_2|w_2)$ 、 $P(x_3|w_2)$) 的计算，所以声学模型问题进一步细化到发射概率 (Emission Probability) 的学习上，可以通过下文的生成式模型 (GMM) 或判别式模型 (DNN) 求解。

另一个问题是，基本单位的粒度大小。对于语音，帧的粒度可通过调节处理窗口的宽窄来控制。对于文本，字词级别的粒度过于宽泛笼统，于是我们往下分解，如图 2.1 所示：字词是由音素 (Phone) 组成的；音素的上下文不同，同一个音素就有了不同的变异体，比如 /l-d-sil/ 与 /u-d-l/ 是一对亲兄弟却是两家子，记为三音素 (Triphone)；每个三音素又可以用一个独立的三状态 HMM 表示，这样，文本方面的基本单位降解为微小的 HMM 状态。由于很多三音素并未在语料中出现或数量不多，并且可以通过决策树 (Decision Tree) 共享三音素的状态，所以对于共有 N 个音素的语言，最终保留下来的三音素状态数量远小于 $3N^3$ ，一般为几千，并把他们叫做 Senones，此时，发射概率记为 $P(x_i|s_j)$ ，其中 s_j 表示第 j 个 Senone，与之对应的帧 (x_i) 的跨度通常取为 25 ms，帧间步移常取为 10 ms，由于跨度大于步移，相邻帧的信息是冗余的，这是跳帧训练和解码的一个出发点。

逐层分解一件事物直至根本，把握住每个关键节点之后，拼装回去，整体复又呈现在眼前，只是理解得更透彻了，这正是还原论 (Reductionism) 和第一性原理思考法 (First Principles Thinking) 的思想。但凡问题，都有其最原始的症结；但凡事物，都有其最基本的要素。语音识别系统的设计和训练正是一个从大到小、从宏观到微观的拆解过程，而语音识别系统的解码是要回去的：从 Frame 到 Senone，从 Senone 到 Triphone，再到 Phone，最后到 Word 直至 Sentence。



HMM 涉及的主要内容有，两组序列（隐含状态和观测值），三种概率（初始状态概率，状态转移概率，发射概率），和三个基本问题（产生观测序列的概率计算，最佳隐含状态序列的解码，模型本身的训练），以及这三个问题的常用算法（前向或后向算法，Viterbi 算法，EM 算法）。语音识别的最终应用对应的是解码问题，所以对语音识别系统的评估也叫做解码 (Decoding)。

2.2.2 GMM-HMM

HMM 确定了语音识别的整体框架，其中发射概率的求取直接取决于声学模型的好坏，也是研究者探索最多的地方。

高斯混合模型（Gaussian Mixture Model, GMM）是最常用的统计模型，给定充分的子高斯数，GMM 可以拟合任意的概率分布，自我迭代式的 EM 算法使得 GMM 的训练较为容易实现，所以 GMM 成为首选的发射概率模型。每个 GMM 对应一个 Senone，并用各自的概率密度函数（Probability Density Function, PDF）表示。GMM 把每帧看成空间中一个孤立的点，点与点之间没有依赖关系，所以 GMM 忽略了语音信号中的时序信息，并且习惯使用帧间相关性较小的 MFCC（Mel Frequency Cepstral Coefficient）特征。

GMM 训练完成后，通过比对每个 PDF，可以求出每个发射概率 $P(x_i|s_j)$ ，然后往上回溯，直到得到句子，这其中会有一系列条件限制，比如，这一串 Senones 能否组成 Triphone，这一串 Triphones 能否组成 Phone，这一串 Phones 能否组成 Word，这一串 Words 能否组成 Sentence，以及组合过程当中，这种选择是否是当下最优的，这些问题可借助加权有限状态转换器（Weighted Finite State Transducer, WFST）统一进行最优路径搜索 [1]。

2.2.3 DNN-HMM

GMM 是生成式模型（Generative Model），着重刻画数据的内在分布，可直接求解 $P(x_i|s_j)$ ，而 $P(x_i|s_j) = P(s_i|x_j)P(x_j)/P(s_j)$ ，因 $P(x_j)$ 省去不算， $P(s_j)$ 可通过常规统计方法求出，问题进一步归结为求取 $P(s_i|x_j)$ ，这是典型的分类（Classification）问题，也是判别式模型（Discriminative Model）所擅长的，其中深度神经网络（Deep Neural Network, DNN）的研究在当下很是繁荣。上述各项也有各自的叫法， $P(x_i|s_j)$ 是似然（Likelihood）， $P(s_j)$ 是先验概率（Prior Probability）， $P(s_i|x_j)$ 是后验概率（Posterior Probability）。

DNN 用于分类问题，是有监督学习（Supervised Learning），标签（Label）的准备是必不可少的。由于训练集中只提供了整条语音与整条文本之间的对应关系，并未明确指出帧级别的标签，所以还需要额外的算法对数据集进行打标签，选择的方法是上文的 GMM。作为生成式模型的 GMM 擅长捕捉已知数据中的内在关系，能够很好地刻画数据的分布，打出的标签具有较高的可信度，但对于未知数据的分类，判别式模型的 DNN 有着更强的泛化能力。通俗点来说，GMM 善于就已有资源进行最大化的开发（Exploitation），而 DNN 擅长举一反三，具有探索精神（Exploration），DNN-HMM 能够超越 GMM-HMM 正是两大态度的强强结合，所以青（DNN）出于蓝（GMM）也就不足为奇了。

相较于 GMM-HMM 结构，DNN-HMM 与之唯一的不同是结构中的发射概率是由 DNN 而非 GMM 求出的，即二者的区别在于 GMM 与 DNN 之间的相互替代。此外，GMM 和 DNN 中的前向神经网络（Feedforward Neural Network），是独立对待各帧的，即上一帧计算的结果不会影响下一帧的计算，忽略了帧与帧之间的时序信息。DNN 起用循环神经网络（Recurrent Neural Network, RNN）时，便可以考虑时序信息了。



贝叶斯定理（Bayes' theorem）已被用到两次，宏观的一次是分出了声学模型和语言模型，微观的一次是构造了 HMM 发射概率的判别式求法。

2.2.4 End-to-End

由于语音与文本的多变性，刚开始的时候我们否决了从语音到文本端到端映射的想法，经过了抽丝剥茧、以小见大，再回过头来看这个问题。假设输入是一整段语音（以帧为基本单位），输出是对应的文本（以音素或字词为基本单位），两端数据都处理成规整的数学表示形式了，只要数据是足够的，选的算法是合适的，兴许能训练出一个好的端对端模型，于是所有的压力就转移到模型上来了，怎样选择一个内心强大的模型是关键。深度学习方法是端对端学习的主要途径。

端对端学习需要考虑的首要问题也是输入输出的不定长问题。

对于输入，可以考虑将不同长度的数据转化为固定维度的向量。如果输入一股脑地进入模型，可以选择使用卷积神经网络（Convolutional Neural Network，CNN）进行转换，CNN 通过控制池化层（Pooling Layer）的尺度来保证不同的输入转换后的维度相同；如果输入分帧逐次进入模型，可以使用 RNN，虽然输入是分开进入的，但 RNN 可以将积累的历史信息在最后以固定维度一次性输出。这两个方法常常用于基于注意力（Attention）的网络结构 [2, 3]。

对于输出，往往要参照输入的处理。先考虑输入长度不做处理的情况，此时输出的长度需要与输入保持匹配。因为语音识别中，真实输出的长度远小于输入的长度，可以引入空白标签充数，这是 CTC（Connectionist Temporal Classification）损失函数 [4] 常用的技巧，如果真实输出的长度大于输入的长度，常规 CTC 就不适宜了；另一个情况是，输入只传来一个向量，这正是上段对输入的处理，也正是注意力模型的手段，它根据这个向量解码出一个文本序列（真正实现时，不同时步传来的向量因着当时的注意力权重有所差异和偏重），此时输出的长度便没有了参照，则需要其他机制来判断是否结束输出，比如引入结束符标签，当输出该标签时便结束输出。

当仔细斟酌了输入输出的不定长问题，目前最主流的两个端对端方法也呼之欲出，即上文提到的基于 CTC 损失函数和注意力网络结构的深度学习方法，且二者可以合用。端对端方法将声学模型和语言模型融为一体，简单明了，实施便捷，是当下语音识别的主要方向之一。随着数据量和计算力的增加，端对端方法行之愈加有效，然而这里仍将语音识别系统拆解开来、逐一透视，只因这是真正理解语音识别的必经之路。

上面简述了声学模型各个层次可以使用的数学模型，而数学模型的参数该如何确定才能物尽其用，就是训练的事了，具体实施细节会在后面结合实验详述。语言模型方面没有太多的枝节，常用的方法基于 n 元语法（N-gram Grammar）或 RNN。

3. 语音识别工具

by 阿汤

开源社区大大加速了计算机科学的研究进展，语音识别领域更是如此，其中广泛使用的语音识别开源工具有 CMUSphinx¹、 HTK²、 Kaldi³等，更多语音识别工具可参考维基百科⁴。本书以 Kaldi 的应用为主。

3.1 Kaldi

Kaldi [5] 是语音识别工具中的后起之秀，年轻正是它的优势之一，可以直接汲取前人的经验，吸收当下已成熟的语音技术，没有历史中的摸爬滚打，避免了积重难返的尴尬。清晰的代码结构，完整的 GMM、WFST 实现，大量适合新手的案例教程，繁荣的开源社区，以及更开放的代码许可，使得 Kaldi 吸引了大批用户。

深度学习广泛应用于语音识别后， Kaldi 先后设计了不同的神经网络构架（nnet1、nnet2、nnet3），其中 nnet3 越来越被研究者所接受，相较于其他两种构架， nnet3 采用计算图（Computational Graph）的思路，可以更容易地设计各种类型的网络结构，并支持多任务并行计算，大大缩短训练时间。

后文将利用 Kaldi 部署实验，并对实验中所涉及的相关概念和技术进行梳理。

¹<https://cmusphinx.github.io>

²<http://htk.eng.cam.ac.uk>

³<http://kaldi-asr.org>

⁴https://en.wikipedia.org/wiki/List_of_speech_recognition_software

3.2 深度学习平台

随着深度学习的发展，更先进的计算平台层出不穷⁵。比起 Kaldi 等术业专攻的平台，通用深度学习框架提供各种深度学习技术，并可拓展应用于多种任务，比如语音识别、计算机视觉、自然语言处理等，所以语音识别系统的建立并不局限于某个平台。最为流行深度学习框架有

TensorFlow⁶（Google 首先开发并使用），
PyTorch⁷（Facebook 首先开发并使用），
Caffe2⁸（Facebook 首先开发并使用，已集成至 PyTorch），
CNTK⁹（Microsoft 首先开发并使用），
MXNet¹⁰（Amazon 等使用），

以及对已有框架的进一步封装，比如 Keras¹¹（TensorFlow 已开发相关 API），其他的不一一列举，其中 PyTorch 等使用动态计算图，较适合快速的研究探索，TensorFlow、Caffe2 等较适合高效的产品部署（包括移动端）。

通用深度学习框架的内核语言多为 C++，前端接口语言多支持 Python，这样的搭配使得保持灵活性的同时又不失计算速度。开源工具的更新换代也很快，比如 Theano¹² 已停止维护，有些功成身退的意味，而面对执着于 Lua 语言的 Torch¹³，更多人选择或转移到了 PyTorch。

面对林林总总的深度学习框架，Microsoft 与 Facebook 发起推出 ONNX¹⁴，让用户可以在不同框架之间转换模型。

语音识别系统有着长久的积淀，并形成了完整的流程（从前端语音信号处理，到声学模型和语言模型的训练，再到后端的解码），而深度学习方法较多地作用于声学模型和语言模型部分（或者端对端模型），所以，深度学习框架常与专有的语音识别工具相结合，各取所长，相互弥补，以减少重复劳动、提高研发效率。

各种开源工具应接不暇，然而善假于物而不囿于物，通晓原理，仍是使用工具的基本原则。

⁵https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

⁶<https://www.tensorflow.org>

⁷<http://pytorch.org>

⁸<https://caffe2.ai>

⁹<https://www.microsoft.com/en-us/cognitive-toolkit>

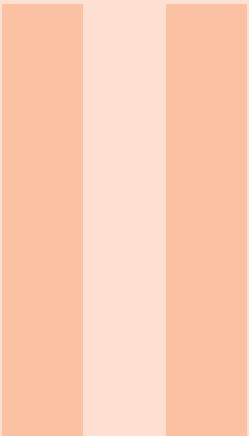
¹⁰<https://mxnet.incubator.apache.org>

¹¹<https://keras.io>

¹²<http://deeplearning.net/software/theano>

¹³<http://torch.ch>

¹⁴<http://onnx.ai>



语音识别基本流程

4	实验先行	23
4.1	代码	
4.2	运行	
4.3	其他案例	
5	前端处理	31
5.1	数据准备	
5.2	声学特征提取	
6	训练与解码	39
6.1	GMM-HMM	
6.2	DNN-HMM	

4. 实验先行

by 阿汤

对语音识别有了大体印象，或许技术流程的层次、细节、原理仍不是太清晰，此时大可以花些时间先行将每个知识点逐一弄明白，等到面面俱到了，这才出山，并期望有了一整套理论支撑便能立马上手一个语音识别系统，而后畅通无阻。这样的线性思维适合一些考试，只要把教材的知识点摸得烂熟，一鼓作气，考个高分就很容易了。只是这种方式并不适用于技术，终究是绝知此事要躬行。有些细节非得实践才能明了，有些经验并非总能见诸文字，更何况技术日新月异，有过时的，有更新的，鱼龙混杂，一劳永逸的方法不可得。显然，对技术该有开放包容的态度和敏锐的感知，时刻准备着更进一层，不断扩大自己的知识领地，所以好的学习策略应是尽快有个自己的知识框架，无论粗糙、精致，首先保证骨架的完整性，然后不断添砖加瓦，一层一层地铺设，每一次都有新面貌、新视野。如果一个方寸一个方寸逐个修葺，这边没完那边便不动工，就无法有的放矢。第二部分从一个完整的语音识别实验开始，走过这一遭，语音识别的基本步骤已于心中有数，随后才是局部的深化和装点。

使用 Thchs30 数据库，可用 Kaldi 快速实现一个基于 DNN-HMM 结构的语音识别系统。本章先行实现该系统的训练和解码，接下来几章简要介绍几大步骤以及每步生成的文件构成。书中实验代码放于 <https://github.com/tzyll/kaldi> 下的 egs/cslt_cases，下文的文件引用将省去这些路径，各子目录中的 steps 和 utils 软链接的是各个案例公用的标准目录，放于 egs/wsj，steps 包含的文件与系统训练和解码步骤直接相关，utils 包含一些可能会用到的实用工具，下文以 steps 和 utils 为开头的文件引用都来自这两个目录，不再表明相对出处。

4.1 代码

参照 README, Linux 下安装好 Kaldi, 进入 asr_baseline (第二部分的实验多以此为当前目录), 其中上层脚本 run.sh 包含了 DNN-HMM 语音识别系统从前期数据准备到最后解码的整个过程, 该脚本是语音识别各个步骤的封装, 每个步骤的脚本又是另一些细枝末节的包裹, 归根结底, 大部分命令会寻至 Kaldi 编译出来的 C++ 可执行程序, 这些 C++ 可执行程序基本放在 kaldi/src 中以 bin 结尾的文件夹中, 其名可望文生义, 从而快速了解功能, 代码本身和 Kaldi 文档¹则有详细介绍。通过 run.sh 中的代码注释可速览整个语音识别的流程, 代码内容如下:

```

1  #!/bin/bash
2
3  # Copyright 2016 Tsinghua University (Author: Dong Wang,
4  #                                     Xuewei Zhang)
5  #                               2018 Tsinghua University (Author: Zhiyuan
6  #                                     Tang)
7  # Apache 2.0.
8
9  . ./cmd.sh ## You'll want to change cmd.sh to something
10   that will work on your system.
11   ## This relates to the queue.
12 . ./path.sh
13
14 n=8 # parallel jobs
15
16
17 ##### Bookmark: basic preparation #####
18
19 # corpus and trans directory
20 thchs=/nfs/public/materials/data/thchs30 Openslr
21
22 # you can obtain the database by uncommenting the following
23   lines
24 # [ -d $thchs ] || mkdir -p $thchs
25 # echo "downloading THCHS30 at $thchs ..."
```

¹<http://kaldi-asr.org/doc>

```
25 # local/download_and_untar.sh $thchs
    http://www.openslr.org/resources/18 data_thchs30
26 # local/download_and_untar.sh $thchs
    http://www.openslr.org/resources/18 resource
27 # local/download_and_untar.sh $thchs
    http://www.openslr.org/resources/18 test-noise
28
29 # generate text, wav.scp, utt2pk, spk2utt in
    data/{train,test}
30 local/thchs-30_data_prep.sh $thchs/data_thchs30
31
32
33 ##### Bookmark: language preparation #####
34
35 # prepare lexicon.txt, extra_questions.txt,
    nonsilence_phones.txt, optional_silence.txt,
    silence_phones.txt
36 # build a large lexicon that involves words in both the
    training and decoding, all in data/dict
37 mkdir -p data/dict;
38 cp $thchs/resource/dict/{extra_questions.txt,
    nonsilence_phones.txt,optional_silence.txt,
    silence_phones.txt} data/dict && \
39 cat $thchs/resource/dict/lexicon.txt
    $thchs/data_thchs30/lm_word/lexicon.txt | \
40 grep -v '<s>' | grep -v '</s>' | sort -u >
    data/dict/lexicon.txt
41
42
43 ##### Bookmark: language processing #####
44
45 # generate language stuff used for training
46 # also lexicon to L_disambig.fst for graph making in
    local/thchs-30_decode.sh
47 mkdir -p data/lang;
48 utils/prepare_lang.sh --position_dependent_phones false
    data/dict "<SPOKEN_NOISE>" data/local/lang data/lang
49
```

```
50 # format trained or provided language model to G.fst
51 # prepare things for graph making in
52   local/thchs-30_decode.sh, not necessary for training
53 (
54   mkdir -p data/graph;
55   gzip -c $thchs/data_thchs30/lm_word/word.3gram.lm >
56     data/graph/word.3gram.lm.gz
57   utils/format_lm.sh data/lang
58     data/graph/word.3gram.lm.gz
59     $thchs/data_thchs30/lm_word/lexicon.txt
60     data/graph/lang
61 )
62
63 ###### Bookmark: feature extraction #####
64
65 # produce MFCC and Fbank features in
66   data/{mfcc,fbank}/{train,test}
67 rm -rf data/mfcc && mkdir -p data/mfcc && cp -r
68   data/{train,test} data/mfcc
69 rm -rf data/fbank && mkdir -p data/fbank && cp -r
70   data/{train,test} data/fbank
71 for x in train test; do
72   # make mfcc and fbank
73   steps/make_mfcc.sh --nj $n --cmd "$train_cmd"
74     data/mfcc/$x
75   steps/make_fbank.sh --nj $n --cmd "$train_cmd"
76     data/fbank/$x
77   # compute cmvn
78   steps/compute_cmvn_stats.sh data/mfcc/$x
79   steps/compute_cmvn_stats.sh data/fbank/$x
80 done
81
82 #####
83 #####
84 ##### Bookmark: GMM-HMM training & decoding #####
85
86 # monophone
87 steps/train_mono.sh --boost-silence 1.25 --nj $n --cmd
```

```
    "$train_cmd" data/mfcc/train data/lang exp/mono
78 # test monophone model
79 local/thchs-30_decode.sh --nj $n "steps/decode.sh"
    exp/mono data/mfcc &
80 # monophone ali
81 steps/align_si.sh --boost-silence 1.25 --nj $n --cmd
    "$train_cmd" data/mfcc/train data/lang exp/mono
    exp/mono_ali

82
83 # triphone
84 steps/train_deltas.sh --boost-silence 1.25 --cmd
    "$train_cmd" 2000 10000 data/mfcc/train data/lang
    exp/mono_ali exp/tri1
85 # test tri1 model
86 local/thchs-30_decode.sh --nj $n "steps/decode.sh"
    exp/tri1 data/mfcc &
87 # triphone_ali
88 steps/align_si.sh --nj $n --cmd "$train_cmd"
    data/mfcc/train data/lang exp/tri1 exp/tri1_ali

89
90 # lda_mllt
91 steps/train_lda_mllt.sh --cmd "$train_cmd" --splice-opts
    "--left-context=3--right-context=3" 2500 15000
    data/mfcc/train data/lang exp/tri1_ali exp/tri2b
92 # test tri2b model
93 local/thchs-30_decode.sh --nj $n "steps/decode.sh"
    exp/tri2b data/mfcc &
94 # lda_mllt_ali
95 steps/align_si.sh --nj $n --cmd "$train_cmd"
    --use-graphs true data/mfcc/train data/lang exp/tri2b
    exp/tri2b_ali

96
97 # sat
98 steps/train_sat.sh --cmd "$train_cmd" 2500 15000
    data/mfcc/train data/lang exp/tri2b_ali exp/tri3b
99 # test tri3b model
100 local/thchs-30_decode.sh --nj $n "steps/decode_fmllr.sh"
    exp/tri3b data/mfcc &
```

```
101 # sat_ali
102 steps/align_fmllr.sh --nj $n --cmd "$train_cmd"
    data/mfcc/train data/lang exp/tri3b exp/tri3b_ali
103
104 # quick
105 steps/train_quick.sh --cmd "$train_cmd" 4200 40000
    data/mfcc/train data/lang exp/tri3b_ali exp/tri4b
106 # test tri4b model
107 local/thchs-30_decode.sh --nj $n "steps/decode_fmllr.sh"
    exp/tri4b data/mfcc &
108 # quick_ali
109 steps/align_fmllr.sh --nj $n --cmd "$train_cmd"
    data/mfcc/train data/lang exp/tri4b exp/tri4b_ali
110
111 ###### Bookmark: DNN training & decoding #####
112
113
114 # train tdnn model
115 tdnn_dir=exp/nnet3/tdnn
116 local/nnet3/run_tdnn.sh data/fbank/train exp/tri4b_ali
    $tdnn_dir
117
118 # decoding
119 graph_dir=exp/tri4b/graph_word # the same as gmm
120 steps/nnet3/decode.sh --nj $n --cmd "$decode_cmd"
    $graph_dir data/fbank/test $tdnn_dir/decode_test_word
121
122 ###### Bookmark: discriminative training & decoding #####
123
124
125 # mmi training
126 criterion=mmi # mmi, mpfe or smbr
127 local/nnet3/run_tdnn_discriminative.sh --criterion
    $criterion $tdnn_dir data/fbank/train
128
129 # decoding
130 steps/nnet3/decode.sh --nj $n --cmd "$decode_cmd"
    $graph_dir data/fbank/test
```

```
131 ${tdnn_dir}}_${criterion}/decode_test_word  
132  
133 exit 0
```

4.2 运行

根据服务器配置，设置 cmd.sh，其中“train_cmd”、“decode_cmd”、“cuda_cmd”分别表示 CPU 训练任务、解码任务、GPU 训练任务可以调用的机器或机器集群。具体地，“run.pl”表示本机运行，“queue.pl”表示使用 Grid Engine²集群，如 OGS/GE³，并通过运行 qconf -sql 查看服务器已有的集群配置文件，并将其作为 queue.pl 的参数。

执行 run.sh（通常后台形式，如 nohup ./run.sh > run_asr.log &），可以从 0 到 1 实现整个语音识别系统的训练和解码，根据日志文件查看实验进度。日志是程序开发的重要文件，可通过阅读日志跟踪、了解系统的运行内容，并可根据日志快速定位、调试错误。Kaldi 每一个关键步骤都会生产日志，存放于输出目录中的 log 文件夹或以 .log 结尾的文件中，这些日志文件是学习和使用 Kaldi 必不可少的材料。

T 执行 ./path.sh，使得当前 shell 下可以直接调用 Kaldi 编译出来的 C++ 可执行程序和相关脚本，方便进一步分析和使用，比如将中断的语句单独取出，并于命令行运行，可加快调试。当不加任何参数直接运行这些程序或脚本时，可打印使用方法。

4.3 其他案例

语音相关的常规任务包括语种识别（Language Recognition）、说话人识别（Speaker Recognition）等，相关案例参见 cslt_cases/{lre_baseline,sre_dvector,sre_ivector}。此外，Kaldi 针对说话人识别提供了 x-vector（egs/sre16）系统。

关于语音识别的端对端学习，相关案例可参见 DeepSpeech⁴、Eesen⁵、ESPnet⁶、wav2letter++⁷ 等。

²<http://wiki.gridengine.info/wiki>

³<http://gridscheduler.sourceforge.net>

⁴<https://github.com.mozilla/DeepSpeech>

⁵<https://github.com/srvk/eesen>

⁶<https://github.com/espnet/espnet>

⁷<https://github.com/facebookresearch/wav2letter>

5. 前端处理

by 阿汤

模型的训练由数据驱动，有些数据是必须要准备的。为了让机器学会将语音转换为文本，首先需要给它提供大量的例子，即语音及其对应的文本，这是原始素材，也最能反映学习目的。这些数据的符号化和结构化则需要一些人类其他知识，包括语言知识和数字信号处理知识。

5.1 数据准备

1) 基本数据

run.sh 中的 `Bookmark: basic preparation` 对 Thchs30 原始数据进行了形式上的处理，以适应 Kaldi 的需要。一个常规的有监督语音识别数据集必然包括一一对应的语音和文本，说话人的信息有好处但非必要。Thchs30 经过初步处理后得到四种文本文件，可以直接打开查看（比如训练集则放在 `data/train` 下），这四个文件也是 Kaldi 的必需文件：

- `wav.scp`, 每条语音的 ID 及其存储地址;
- `text`, 每条语音的 ID 及其对应文本;
- `utt2spk`, 每条语音的 ID 及其说话人 ID;
- `spk2utt`, 每个说话人的 ID 及其所说语音的所有 ID, 使用 `utils/spk2utt_to_utt2spk.pl` 或 `utils/utt2spk_to_spk2utt.pl` 将其与 `utt2spk` 进行相互转换。

对于不同数据源或任务，可能需要另外准备一些文件，比如 `segments` 文件标记每条小语音属于某条大语音的哪一部分，文件格式形如“`<segment-id> <recording-id> <start-time> <end-time>`”，时间以秒计，`extract-segments` 读取此文件对音频进行批量剪切并

保存为 Kaldi 支持的格式（`sox` 也可逐条切割音频）； `spk2gender` 文件标明每个说话人的性别，用于性别识别； `utt2lang` 文件标明每条语音 ID 对应的语种 ID，用于语种识别。由于不同的数据集有着不同的编排，并没有统一的工具提取出以上文件。当根据某个数据集自行生成以上类别的文件并用 `sort` 排序后，可以使用 `utils/validate_data_dir.sh` 校验是否满足 Kaldi 需求，并使用 `utils/fix_data_dir.sh` 进行修复。根据数据校验和修复脚本也可侧面了解 Kaldi 支持的文件类型和格式。



`utt2spk` 和 `spk2utt` 是 Kaldi 处理所必须的，有时候如果不能及时提供说话人信息，可以“伪造”，比如每条语音的说话人 ID 直接使用这条语音的 ID，这对语音识别性能影响不大，但在做说话人识别任务时，显然务必要提供真实的说话人信息。此外，两个文件都需要排序，为保证二者顺序的总体一致性，通常句子 ID 的前端设置为说话人 ID。

2) 语言资料

语言知识方面，Kaldi 至少需要以下文件，见 `Bookmark: language preparation`，存放在 `data/dict` 下：

- `lexicon.txt`，发音词典，即每个词与其所对应的音素串，格式为“`word phone1 phone2 phone3 ...`”；
- `lexiconp.txt`，与 `lexicon.txt` 作用相同，多了发音概率，格式为“`word pronunciation-probability phone1 ...`”，可由系统通过 `lexicon.txt` 自动生成（此时所有词的概率相同），二者提供一个即可，`lexiconp.txt` 优先使用；
- `silence_phones.txt`，静音类音素，包括静音（`sil` 或者 `SIL`）、噪音、笑声等非语言直接相关的伪音素，同一行的音素是某一个音素的不同变体（重音、音调方面），故可共享决策树根；
- `nonsilence_phones.txt`，语言直接相关的真实音素，同一行的音素是某一个音素的不同变体（重音、音调方面），故可共享决策树根；
- `optional_silence.txt`，备用的静音类音素，一般直接来自 `silence_phones.txt` 中的 `sil` 或者 `SIL`；
- `extra_questions.txt`，可为空，同一行的音素有着相同的重音或音调，与 GMM 训练中自动生成的“`questions`”一同用于决策树的生成。

当针对同一种语言，基于新的数据集另起炉灶时，这些文件都可以直接移植过去复用。运行 `egs/wsj/s5/local/wsj_prepare_dict.sh` 可以瞥见如何利用 CMU 英语发音词典¹构建出其他所需文件，Thchs30 中文数据集提供了现成的结果。发音词典应尽可能覆盖训练语料，且基于已有的音素表，可更改或扩充发音词典，以适用于不同的领域或场景。到此为止，Kaldi 用于语音识别系统训练的数据都齐全了，后来的事便是 Kaldi 对这些数据的自动处理和使用。

¹<http://www.speech.cs.cmu.edu/cgi-bin/cmudict>

T 在决策树的生成当中，`nonsilence_phones.txt` 倡“合”，`extra_questions.txt` 促“分”，然而前者中的一行如果由同一基音素衍生出来，具有不同的重音或音调，则在后者中常常处于不同的行，这时保留根源性，即以前者为准。

Bookmark: language processing 中 `utils/prepare_lang.sh` 对 `data/dict` 进行了处理，得到 `data/lang`。选项“`position_dependent_phones`”指明是否使用位置相关的音素，即是否根据一个音素在词中的位置将其加上不同的后缀：“`_B`”（开头）、“`_E`”（结尾）、“`_I`”（中间）、“`_S`”（独立成词）。参数“`<SPOKEN_NOISE>`”取自 `lexicon.txt`，后续处理中所有集外词（Out Of Vocabulary, OOV）都用它来代替。`lang` 中生成的文件：

- `phones.txt`, 将所有音素一一映射为自然数，即音素 ID，引入“`<eps>`”（epsilon）、消歧（Disambiguation）符号“`#n`”（n 为自然数），便于 FST 处理；
- `words.txt`, 将词一一映射为自然数，即词 ID，引入“`<eps>`”（epsilon）、消歧符号“`#0`”、“`<s>`”（句子起始处），“`</s>`”（句子结尾处），便于 FST 处理；
- `oov.txt`, `oov.int`, 集外词的替代者（此处为 `<SPOKEN_NOISE>`）及其在 `words.txt` 中的 ID；
- `topo`, 各个音素 HMM 模型的拓扑图，第二章提过将一个音素（或三音素）表示成一个 HMM，此文件确定了每个音素使用的 HMM 状态数以及转移概率，用于初始化单音素 GMM-HMM，可根据需要自行进行修改（并用 `utils/validate_lang.pl` 校验），实验中静音音素用了5个状态，其他音素用了3个状态。
- `L.fst`, `L_disambig.fst`, 发音词典转换成的 FST，即输入是音素，输出是词，两个 FST 的区别在于后者考虑了消歧。
- `phones/`, 是 `dict/` 的拓展，内部文件均可以文本形式打开查看，后缀为 `txt/int/csl` 的同名文件之间是相互转换的，其中 `context_indep.txt` 标明了上下文无关建模的音素，通常为静音音素，`wdisambig.txt/wdisambig_phones.int/wdisambig_words.int` 分别标明了 `words.txt` 引入的消歧符号（#0）及其在 `phones.txt` 和 `words.txt` 中的 ID，`roots.txt` 定义了同一行音素的各个状态是否共享决策树的根以及是否拆分，对应的音素集则存放在 `sets.txt`。

T 消歧是为了确保发音词典能够得到一个确定性的（Deterministic）WFST。如果有些词的音素串是另一些词的前缀，则需要引入消歧音素加在前者后面。

语言模型方面，可以单独提供 ARPA 格式的统计语言模型，也可以由现有文本训练出来（如使用 Kaldi LM 工具或 SRILM 工具包的 `ngram-count`，具体训练方法可参照 `egs/fisher_swbd/s5/local/fisher_train_lms.sh`），`utils/format_lm.sh` 将该语言模型转换为 `G.fst`，即输入是词，输出也是词，与 `data/lang` 中的文件一同放在 `data/graph/lang` 下，用于后面制作解码图，与模型的训练无关。

5.2 声学特征提取

原始音频信号可以直接作为模型的输入，只是在保守情况下，如数据不足、计算力薄

弱时，更讨好的做法是先将其由时域信号转换为频域信号，借鉴人耳的处理机制，最终产生声学特征（Acoustic Feature）。声学特征提取使得语音信息更容易暴露，大大降低算法优化的压力，某种程度上也起到降维的效果，提高计算效率，比如 16 kHz 下的 25 ms 共 400 个数值可转换为 40 维的声学特征。

Bookmark: feature extraction 分别提取音频的 MFCC（Mel Frequency Cepstral Coefficient，梅尔频率倒谱系数）和 FBANK（Mel Filter Bank，梅尔滤波器组）两种声学特征，并计算二者关于说话人的倒谱均值和方差统计量，用于 CMVN（Cepstral Mean and Variance Normalization）标准化。

计算 MFCC 和 FBANK 之前需要通过 `conf/{mfcc.conf,fbank.conf}` 设置相关选项，可通过命令行运行 `compute-mfcc-feats` 和 `compute-fbank-feats` 得知可设置哪些选项，每个选项基本都有合理的默认值。原理上 MFCC 是基于 FBANK 生成的，Kaldi 则将两种特征的计算过程分别包装成两个命令。MFCC 特征各维度之间具有较弱的相关性，适合 GMM 的训练，FBANK 特征用于 DNN 的训练则更有优势。两种特征需要注意的选项有：

- `sample-frequency`，音频的采样频率，默认为 16000 (16 kHz)，如果真实数据与此不同需要指明；
- `num-mel-bins`，梅尔滤波器个数，两种特征兼有，对于 FBANK 来说也是特征维度（通常设为 40）；
- `num-ceps`，MFCC 特征专有，倒谱个数，也是特征的维度，须不大于 `num-mel-bins`，通常使用默认值 13，首维较为特殊，为第 0 个倒谱系数，即 C_0 ，如果 `use-energy` 设为 `true`，则首维替换为能量值（对数形式）；
- `use-energy`，对于 MFCC 来说，表明计算时是否使用“energy”，GMM-HMM 的训练通常将其设置为“`false`”，对于 FBANK 来说，表明计算时是否在特征首部多加一维能量值，`compute-vad` 使用 MFCC 或 FBANK 中的能量值用于语音活动检测（Voice Activity Detection，VAD）时，此选项须设为“`true`”。

实验中对于训练集，MFCC 和 FBANK 以及它们的均值和方差统计量分别存放在 `data/mfcc/train` 和 `data/fbank/train` 的 `data` 文件夹下，以“`.ark`”（archive，存档文件）和“`.scp`”（script，前者的索引文件）格式同步存在，最终将所有索引文件合并放入上一级目录中，即 `feats.scp` 和 `cmvn.scp`。通过索引文件可以找到每条语音特征数据的存放位置，并可通过 `copy-feats` 以文本形式打印出来，每条语音的声学特征都以矩阵形式存储，每一行为一帧，宽度即为特征维度，可用 `feat-to-dim` 查看，行数即为帧数，不同长度的语音有着不同的帧数，可用 `feat-to-len` 查看每条或所有语音的总帧数（考虑到每帧的跨度和步移，有助于估算语音的总时长，大体等于总帧数与步移之积）。



Kaldi 以 `ark` 和 `scp` 两种格式同步存储文件，相关命令对这些数据进行读写（I/O）时需要特别注明，并可添加相关选项，下面是基于 `copy-feats` 的几个例子。

1. 读 `ark`，写文本 `ark`:

```
copy-feats ark:raw_mfcc_train.1.ark ark,t:tmp.ark
```

2. 读 `scp`，写文本 `ark`:

```
copy-feats scp:feats.scp ark,t:tmp.ark
3. 读 scp, 写二进制 ark:
copy-feats scp:feats.scp ark:tmp.ark
4. 读 ark, 写二进制 ark、scp:
copy-feats ark:raw_mfcc_train.1.ark ark,scp:tmp.ark,tmp.scp
5. 读写 scp 时, 忽略数据丢失错误:
copy-feats scp,p:feats.scp ark,scp,p:tmp.ark,tmp.scp
另外, 同时写 ark 和 scp 时 ark 须放在 scp 前面, scp 不能单独写入; 选项相对于 ark/scp 的位置不分先后; 多种选项可以同时存在。
```

提取声学特征的目的是在保证音素可辨的情况下, 增强信号对说话人、噪声、信道等的鲁棒性, 常用的声学特征有 FBANK、MFCC、PLP (Perceptual Linear Prediction) 等。Kaldi 中使用 `compute-plp-feats` 提取 PLP 特征。下面以 MFCC 特征为例说明其产生的主要步骤。声学特征提取需要预先对音频做一些处理, 并将其转换至频域, 进一步产生 FBANK 和 MFCC 特征, 综合起来, 大体上主要有如图 5.1 所示的流程。

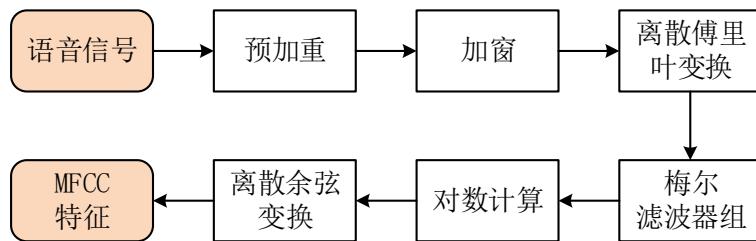


图 5.1: MFCC 特征的提取流程

5.2.1 预加重 (Pre-emphasis)

语音中有频谱倾斜 (Spectral Tilt) 现象, 即低频具有较高能量, 因此, 需要加重高频语音的能量, 使得高频信息凸显出来, 其计算方法为

$$x'[t] = x[t] - \alpha x[t-1] \quad (5.1)$$

其中, $x[t]$ 表示音频数据 (可以看成一个向量) 的第 t 个数, α 通常取值范围是 $(0.95, 0.99)$ 。

预加重之前也可先对音频进行抖动 (Dithering), 抖动是信号处理常用的手段, 它将信号加入低剂量随机噪音, 可有效降低录制音频时模数转换产生的量化误差 (Quantization Error), 似一种以其人之道还治其人之身的方法。

5.2.2 加窗 (Windowing)

特征提取时, 如果每次取出 25 ms (跨度) 的语音, 进行离散傅里叶变换计算出一帧, 接着步移 10 ms 继续计算下一帧, 这种基本做法就是矩形窗 (Rectangular Window), 棱角分明的窗容易造成频谱泄露 (Spectral Leakage), 可以选择使用钟形窗, 如海明窗 (Hamming Window)、汉宁窗 (Hanning Window) 等。加窗的计算方法为

$$x'[n] = w[n]x[n] \quad (5.2)$$

其中 $x[n]$ 是所取窗口（窗长为 N ）之内的第 n 个数， $w[n]$ 是与之对应的权重，不同的加窗方式则体现在 w 的取值上，以 $N = 400$ ($16\text{ kHz} * 25\text{ ms}$) 为例，图 5.2 展示了不同窗口的形状。本质上，加窗计算也是卷积（Convolution）。

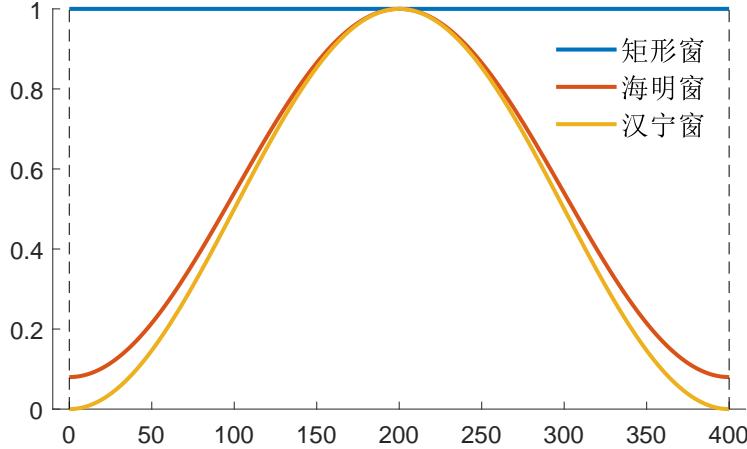


图 5.2: 不同窗口的形状（从上到下分别为矩形窗、海明窗、汉宁窗）

5.2.3 离散傅里叶变换 (DFT)

DFT 从每一段加窗后的音频中分别提取出频域信息，计算方式为

$$X[k] = \sum_{n=0}^{N-1} x[n] \exp\left(-j \frac{2\pi}{N} kn\right), \quad 0 \leq k < N \quad (5.3)$$

通过复数 $X[k]$ 可计算第 k 个频段的幅度（Magnitude）和相位（Phase），幅度之于频率的坐标图即是频谱（Spectrum），一段语音的所有频谱按时间顺序横排在一起便是这段语音的频谱图（Spectrogram），如之前的图 1.4，每一列都是一个频谱（颜色明暗表示数值大小）。

DFT 的一个实现方法是快速傅里叶变换（Fast Fourier Transform, FFT），可将时间复杂度从 $O(N^2)$ 降为 $O(N \log_2 N)$ ，但需要保证窗长 N 是 2 的指数，如果原窗长不满足此条件，一般音频信号 x 补充尾零，如 400 的窗长可扩展为 512。

频谱的具体使用中，通常用 $|X[k]|^2$ 表示第 k 个频段的能量值（忽略了相位信息），记为 Power Spectrum（既是能量频谱，也是频谱的幂），并根据奈奎斯特频率（Nyquist Frequency），只取其前半段（比如 512 的频数，取其前 $512 * 1/2 + 1$ ）作为最终输出结果。声学特征多基于频谱提取出来，甚至就使用频谱本身，`compute-spectrogram-feats` 可用于频谱特征的提取。

5.2.4 FBANK 特征

人耳对不同频率的感知程度不一样，频率越高，敏感度较低，所以人耳的频域感知是非线性的，梅尔刻度（Mel Scale）正是刻画这种规律的，它反映了人耳线性感知的梅尔频率（Mel Frequency）与一般频率之间的关系，梅尔频率 $Mel(f)$ 与一般频率 f 的转换公

式为

$$Mel(f) = 1127 \ln\left(1 + \frac{f}{700}\right) \quad (5.4)$$

将频谱规划到梅尔刻度上，能有效促进语音识别系统的性能，实现方法是梅尔滤波器组（Mel Filter Bank）。具体地，将上一节输出的能量频谱通过如图 5.3 所示的三角滤波器组（Triangular Filter Bank）得到梅尔频谱，计算方式与加窗类似，越往高频，滤波器窗口越大，窗口扩大的量级则与梅尔刻度一致。滤波器的个数就是梅尔频段的总数目，通常取为几十。梅尔频谱的能量数值取对数，最终得到的结果就是常说的 FBANK 特征。对数计算增强了特征的鲁棒性，且人类对能量强弱的感知是符合对数关系的。用于 DNN 训练时，FBANK 的维度就是梅尔滤波器的个数，常取 20 到 40 之间。

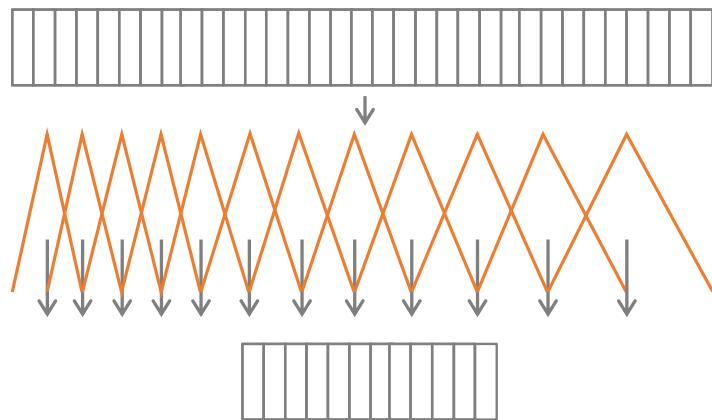


图 5.3: 三角滤波器组的工作方式

T 梅尔滤波器的计算方式与加窗类似，只是窗口跨度依次变大，也属于卷积运算，故可使用卷积神经网络自动学习更加合理的滤波器组。

5.2.5 MFCC 特征

1) 倒谱 (Cepstrum)

FBANK 中含有基频 (Fundamental Frequency, F0) 的谐波 (Harmonic)，可简单理解为频谱中的毛刺，不利于整体轮廓即包络 (Envelope) 的显现，且各维度之间具有较高的相关性，不适宜 GMM 学习。生理上，谐波源于声带 (Vocal Cord)，而真正对具体音素进行调节的是声道 (Vocal Tract)，MFCC 的目的便是消除与音素判别关系不大的谐波，保留包络信息。FBANK 特征进行离散傅里叶反变换 (Inverse Discrete Fourier Transform, IDFT) 可以将包络与谐波分开，具体等价于离散余弦变换 (Discrete Cosine Transform, DCT)，生成结果记为倒谱 (Cepstrum 由 Spectrum 字母倒换而来)，倒谱的低段位系数 (通常前 13 个) 可以描述频谱包络，即梅尔频率倒谱系数。Spectrum 是频域，相对地 Cepstrum 则是时域。

如第 5.2 小结所述，使用 `compute-mfcc-feats` 计算 MFCC 时，如需添加一维能量 (Energy) 值，则由该帧下所有音频采样点取值的平方和计算而来（常取对数），并替换 MFCC 的第一个系数 (C0)，这样 MFCC 特征则是 Energy + 12 MFCCs。



声学特征提取中多次使用到对数计算，对数计算在其他地方也有很多好处，比如它可以一定程度地增加非线性，平滑数据；缩小数据范围，防止溢出；将乘（除）变成加法，方便计算；与 Softmax（神经网络的一种激活函数）合用便于梯度计算和传递等。

2) 动态特征 (Dynamic Feature)

语音是时序信号，故声学特征的帧与帧之间并不是孤立的，是连续变化的，前后的变化往往包含一些声音线索，动态特性可以显示特征随时间变化的程度，常采用一阶导数、二阶导数，一阶导数的简单估算方法为

$$d[t] = \frac{c[t+1] - c[t-1]}{2} \quad (5.5)$$

其中， $c[t]$ 表示第 t 帧 MFCC 特征，二阶偏导也照此法，`add-deltas` 可以完成计算。所以通常用来训练 GMM 的声学特征共 39 维（ Δ 表示 delta）：

12 MFCCs + Energy;

12 Δ MFCCs + Δ Energy;

12 Δ^2 MFCCs + Δ^2 Energy。

最后，Kaldi 通过 `compute-fbank-feats` 和 `compute-mfcc-feats` 综合以上过程分别计算 FBANK 和 MFCC 特征，并通过设置相关选项调节各个步骤的计算，比如“preemphasis-coefficient” 选项设置预加重系数，“window-type” 选项设置加窗类型。

6. 训练与解码

by 阿汤

上文花了一些笔墨来介绍前端数据准备的过程，其一是为了尽快满足 Kaldi 入口的需求，以促成一个语音识别系统；其二，当 Kaldi 与其他深度学习平台协作用于语音识别时，前端声学特征提取常常依赖于 Kaldi；再者，即使不使用 Kaldi 来搭建语音识别系统，常规声学特征（如 FBANK）提取的基本流程是大同小异的，所以，首先掌握语音数据的前端处理，将语音信号转换为标准数学形式（即向量、矩阵）的声学特征，并把主要精力集中在模型的设计和训练上，必然有助于语音识别系统的快速迭代。

语音识别研究的重心终究是模型的设计、训练和解码。无论是 GMM-HMM 还是 DNN-HMM，都是枝繁叶茂，均可独立成册，而且 DNN 本身即是一大方向，无法简单概括，所以以下只简要介绍 Kaldi 中模型训练和解码的基本实施过程。

6.1 GMM-HMM

帧级别的声学特征准备好了，对应的音素串标签可通过查询发音词典将文本转换而来，接下来便是 GMM-HMM 的训练。

6.1.1 训练

GMM-HMM 训练中 HMM 的转移概率也可更新，但参数量和计算量远小于 GMM，所以接下来的 `Bookmark: GMM-HMM training & decoding` 主要进行 GMM 参数的训练。第 2.2 小节提过，每个音素（或三音素）用一个 HMM 建模，每个 HMM 状态的发射概率对应一个 GMM，所以，通俗点说，GMM-HMM 的目的即是找到每一帧属于哪个音素的那个状态。GMM-HMM 的训练使用自我迭代式的 EM 算法，每一次 EM 后都比自己进步一点点；接下来新一代 GMM-HMM 继承父辈的成果（上一代 GMM-HMM 标记的数据），从

头开始学习，青出于蓝，每一代都比上一代进步一大截。这是 GMM 的进化史，每一代都发挥自己的最大潜力，然后把基业交给更具潜力的下一代：训练模型，测试模型性能（对训练来说不是必须的），用模型标记数据，完成一个生命周期，再用标记的数据训练新的模型，循环往复下去。所谓“标记”，是指现阶段模型自以为是的哪一帧属于哪一个音素的哪个状态。实验中经历了 5 个回合。

模型训练的主要目标是调整每个 GMM 的参数，使其能够更好地刻画它所对应的音素的某一状态，所以需要先获取属于该音素状态的所有语音特征，EM 算法基于这些数据做最大似然估计（Maximum Likelihood Estimation, MLE），亦步亦趋地优化 GMM 的 PDF。不同的 GMM 之间是分开进行的，但可使用高斯模型共享等参数共享方法。因为只知道整段语音对应哪个音素串，小粒度上的对应无从知晓，问题便归结为如何获取属于每个 PDF 的所有语音特征。对于第一个开荒者（即 monophone GMM）来说，采用平启动（Flat Start）的方法，即对于一条语音来说，按对应音素串所有的 PDF 数平分语音特征，每个片段的特征归为对应位置的 PDF 所有，当然也得考虑静音类音素。对于继承者来说，直接使用上一个 GMM-HMM 系统强制对齐（Force Alignment）的结果，即标记。强制对齐时，每帧语音特征与每个 PDF 比对，可知它属于每个音素状态的可能性，每帧数据大可自顾自地选择相对概率最大的那个，但须保证有意义，即最终能够通过 HMM 串联回到参照文本，满足这个“强制”条件的情况下，虽然某些帧选择的不是概率最大的 PDF，但总体上仍可保证有个尽可能高的分数，此时便是关于参考文本的似然最大。

不同时期的 GMM 叠加了不同的特性，“mono”基于单音素，“mono”是“monophone”，“tri1”引入三音素，“tri”是“triphone”，“tri2b”对 MFCC 特征进行了基于线性判别分析（Linear Discriminant Analysis, LDA）和最大似然线性变换（Maximum Likelihood Linear Transform, MLLT）的转换，“tri3b”进行了说话人适应（Speaker Adaptation），“tri4b”返璞归真，再将训练好好来过一次。可以使用 `gmm-info` 查看不同时期 GMM-HMM 系统的具体信息。

6.1.2 解码

如第 2.2 所述，GMM 训练完成后，通过比对每个 PDF，可以求出每个发射概率 $P(x_i|s_j)$ ，然后往上回溯，直到得到句子，这其中会有一系列条件限制，比如，这一串 Senones 能否组成 Triphone，这一串 Triphones 能否组成 Phone，这一串 Phones 能否组成 Word，这一串 Words 能否组成 Sentence，以及组合过程当中，这种选择是否是当下最优的，这些问题可借助加权有限状态转换器（Weighted Finite State Transducer, WFST）统一进行最优路径搜索 [1]。

测试模型性能就是看其解码效果，并由识别文本相对于参照文本的词错误率（Word Error Rate, WER）来衡量，其计算方式是 $WER = \frac{S+D+I}{N}$ ，其中 $S+D+I$ 是编辑距离，包括三种错误：替换（Substitution）、删除（Deletion）和插入（Insertion）， N 是参照文本的总词数。解码需要解码图，即加权有限状态转换器，`local/thchs-30_decode.sh` 中 `Bookmark: graph making` 生成 `HCLG.fst`，具体放在 GMM 生成目录下的图目录里（本实验是 `graph_word`），接着可以看到，`Bookmark: GMM decoding` 调用 `HCLG.fst` 进行最优路

径搜索，即解码。因为解码脚本默认在解码生成结果所在目录的上一层目录寻找 GMM 模型，所以解码目录有必要放在 GMM 目录之下（下午的 DNN 解码亦是如此）。HCLG.fst 中，H (H.fst) 的输入是 transition-id (Kaldi 所定义，PDF 及相关转移操作所对应的 ID)，输出是三音素，包含了 HMM 信息，C (C.fst) 的输入是三音素，输出是音素，包好了音素的上下文关系，L (L.fst)、G (G.fst) 已在第 5.1 小节准备妥当，分别包含了词典和语法信息。解码图可以完成从 Senone 到 Triphone，再到 Phone，最后到 Word 直至 Sentence 的最优路径搜索。

6.1.3 强制对齐

用模型标记数据就是上面说的强制对齐，即每帧对应哪个 PDF。对齐的结果存放在 exp/*_ali 中，以 mono_ali 为例，用 `gunzip -c ali.1.gz | less` 打印出来，每句话都有一串整数，每个整数是相应位置的帧所属 transition-id，为了将每帧对应到 pdf-id，可以通过 `gunzip -c ali.1.gz | ali-to-pdf final.mdl ark:- ark,t: | less`，并可通过 `gunzip -c ali.1.gz | ali-to-phones final.mdl ark:- ark,t: | less` 将每帧对应到 phone-id。



Kaldi 定义了一些操作对象，并用整数表示其 ID，这是最基本的符号化：

1. phone-id：音素的 ID，参见 data/lang/phones.txt，强制对齐的结果不含 0 (表示<eps>) 和消歧符 ID；
2. hmm-state-id，单个 HMM 的状态 ID，从 0 开始的几个数，参见 data/lang/topo；
3. pdf-id，GMM 的 ID，从 0 开始，总数确定了 DNN 输出节点数，通常有数千个；
4. transition-index，标识一个状态的不同转移，从 0 开始的几个数；
5. transition-id，上面四项的组合 (phone-id,hmm-state-id, pdf-id, transition-index)，可以涵盖所有可能动作，表示哪个 phone 的哪个 state 的哪个 transition 以及这个 state 对应的 pdf 和这个 transition 的概率，其中元组 (phone-id,hmm-state-id, pdf-id) 单独拿出来，叫 transition-state，与 transition-id 都从 1 开始计数。

GMM-HMM 是上一代语音识别系统结构，已有几十年积淀，技术较为成熟和稳定，Kaldi 对其做了极好的流程化整合，所以在后期研究中多对 GMM-HMM 做着老实的搬运，不花多少心思，但这套算法传达的思想源远流长，将影响语音领域多年。

6.2 DNN-HMM

一个复杂的功能指望一个复杂的函数来实现，可以找张纸洋洋洒洒地写上一长串复杂的多项式计算，只是这加减乘除、这对数指数、这括号该如何排列组合不好盘算，还不如画个图来得直截了当，于是 DNN 某个层面的基本思想就是将各种嵌套、组合的多项式计算以图的形式可视化出来，记为“计算图”(Computational Graph)。所见即所得，计算图更契合人的认知感受，想象空间里也多了质地感。从公式到图，像是从算盘到画布，从线性到立体，换了个视角，所以“图”的作用也只是增加了函数设计的深度和广度，降低了难度，本质上的数学计算从未变过。而且，DNN 的基本算子很普通：权重连接 (Weight)、

激活函数（Activation），好比是庞大的计算机体系最原始的形态只是简单地 0/1，小的行为相互影响、触动，众志成城，积沙成塔，才涌现了 DNN 的“大智慧”。

6.2.1 主要过程

最后一代的 GMM-HMM 将数据打完标签后，递交给 DNN 进行有监督学习，训练过程是典型的分类器（Classifier）学习。

Bookmark: DNN training & decoding 中训练了一个时延神经网络（Time Delay Neural Network, TDNN）[6]。DNN 的作用是替代了 GMM 来计算 HMM 的发射概率，并不影响 HMM（H.fst），更不用说音素（C.fst）、发音词典（L.fst）和语言模型（G.fst），所以解码时仍然沿用了 GMM-HMM 的 HCLG.fst。

可在 conf/decode_dnn.config 中配置解码参数，其中常用的选项为 beam 和 lattice-beam。beam 是解码时集束搜索（Beam Search）的宽度，宽度越大，搜索结果越准确，相应地搜索时间越长；lattice-beam 是生成 Lattice（简单理解为包含了很多可能的解码路径，留待进一步择优）的宽度，宽度越大，Lattice 越大，相应地搜索时间越长。

语音识别系统的最终衡量标准是 WER，是序列（Sequence）上的尺度上，而 GMM 以最大似然为目标，DNN 以最小化交叉熵（Cross Entropy, CE）或均方误差（Mean Squared Error, MSE）等为损失函数，二者训练时的目标都在帧的尺度上，造成训练与推断（Reference）的不一致，所以可以引入更靠近序列层面的训练准则，也就是序列判别训练（Sequence-Discriminative Training, SDT），常用的准则有：MMI（Maximum Mutual Information）和 BMMI（Boosted MMI），MPE（Minimum Phone Error），sMBR(state Minimum Bayes Risk)。SDT 最开始用于 GMM-HMM，后来移植到 DNN-HMM 上，**Bookmark: discriminative training & decoding** 使用了 MMI，由于模型本身结构不变，只改变了训练的准则，解码方式仍与之前相同。

6.2.2 nnet3 配置

Kaldi nnet3 的神经网络结构是通过读取包含相关组件信息的配置文件生成的。配置文件由实验人员直接编写，它描述了网络中各个组件的拓扑关系。以下简介了 nnet3 配置文件的基本语法，为“绘制”不同的神经网络结构作参考。

Kaldi nnet3 参照的仍是计算图的思想，即将神经网络看成是由多个不同计算单元或步骤按特定顺序连接起来的图，并对该计算图进行编译和执行。nnet3 的配置文件则是对图的构造进行详细的描述，用于网络的初始化，比如含有一层隐含层（激活函数为 Rectifier）的前向神经网络可以描述为：

```
# First the components
component name=affine1 type=AffineComponent input-dim=30 output-dim=1000
component name=relu1 type=RectifiedLinearComponent dim=1000
component name=affine2 type=AffineComponent input-dim=1000 output-dim=800
component name=logsoftmax type=LogSoftmaxComponent dim=800
# Next the component-nodes
```

```

input-node name=input dim=10
component-node name=affine1_node component=affine1 input=Append( Offset(input, -2),
Offset(input, 0), Offset(input, 1))
component-node name=nonlin1 component=relu1 input=affine1_node
component-node name=affine2_node component=affine2 input=nonlin1
component-node name=output_nonlin component=logsoftmax input=affine2_node
output-node name=output input=output_nonlin objective=quadratic

```

1) component 和 component-node

上例中可以看出，配置文件分为两大部分：component 和 component-node，它们之间的关系可以类比于“类的实例化”。component 可以看作是类，它定义了网络中所有情形的组件，每个类（component）都有独一无二的名称，属性的设置相互不受干扰，属性包括类型（权重连接方式和不同的激活函数）、输入输出的维度，以及该类的特性。component-node 则是对 component 的实例化，在对它进行定义时，它所属的类由后面的 component 选项给出，并且需要明确它的输入（另一个 component-node）。比如上例中，component-node affine2_node 实例化了 component affine2，并且指定了它的输入为 component-node nonlin1，nonlin1 的维度与 affine2 所要求的输入维度是相同的。

input-node 和 output-node 是特殊的标识符，不需要指定类（component），它们分别代表神经网络的输入和输出节点，可以有多个（名称不同）。output-node 中可以通过 objective 选项指定不同的损失函数，如 linear（交叉熵，不指定 objective 选项时的默认设置）、quadratic（均方误差）。还有一个不需要类的特殊标识符 dim-range-node，它用于截取一个节点的部分作为输入，比如我们可以通过以下语句截取 node1 传入值（向量）的前100维，并赋值给 node2：

```
dim-range-node name=node2 input-node=node1 dim-offset=0 dim=100
```

一个 component 可以对应多个不同名 component-node，此时，这些 component-node 也将共享相同的参数。component 可以不被实例化，component-node 必须要有定义了的 component。

2) 属性与描述符

Kaldi 中实现了多种常见的和定制化的 component，具体名称、作用和属性，可参考源文件 nnet3/nnet-{simple,normalize,convolutional,attention,combined,general}-component.h，其中的常规方法 InitFromConfig 表示了网络在初始化过程中需要从配置文件中读取的信息，常见的属性有 type（直接使用类名，如激活函数 SigmoidComponent、RectifiedLinearComponent，全连接 AffineComponent、NaturalGradientAffineComponent 等），input-dim 和 output-dim（对于输入输出维度相同的组件来说，直接用 dim 表示二者），还有一些组件有自己特定的属性，比如卷积层 ConvolutionComponent 需要指定滤镜的大小、步长和个数。

描述符	功能
node1	最基本形式，没有修饰语， node1 的值直接传入。
Append(node1,node2,...)	将 node1、node2 等组件传入的值拼接起来，结果的维度是各组件维度之和。
Sum(node1,node2)	将 node1、node2 传入的值加起来， node1、node2 以及最终结果的维度相同。
Failover(node1,node2)	如果 node1 是不可计算的，则使用 node2。
IfDefined(node1)	如果 node1 目前还没有计算出来，先以“0”值代替。
Offset(node1,value)	node1 传过来的值是相对于当前时标偏移 value（可正可负）时步时的结果。
Switch(node1,node2,...)	按顺序，每个时步使用其中一个 node，该 node 的序号是当前时步对 node 总数取模的结果。
Scale(value,node1)	node1 传入的值乘以系数 value。
Const(value,dim)	维度为 dim、各元素取值为 value 的常值向量。
Round(node1,value)	每个时步，将 node1 的时标（time-index）设置为小于当前时步且又是 value 的最大倍数。
ReplaceIndex(node1,t,value)	每个时步，node1 的时标保持不变，恒为 value。t 也可换为其他变量。

表 6.1: Kaldi nnet3 配置文件中的描述符

上例中出现的其他关键字（Append、Offset）是 nnet3 的描述符（Descriptor），出现于 component-node 的声明中，由源文件 nnet3/nnet-descriptor.h 定义。各个组件相互粘合时，描述符常常用于 input 选项的量化说明，比如上例中 affine1_node 的输入是由 3 个不同时步（time-step）的原始输入（input-node）拼接（Append）而成的，而 Offset 则表示了某个时步的输入与当前时步的相对时间偏移。如果上例用于语音识别任务，affine1_node 的输入则是由当前帧的上上帧、当前帧以及当前帧的下一帧拼接而成，共有 $10 * 3 = 30$ 维。描述符的使用方法为： `input=[Descriptor]`，表 6.1 总结了 nnet3 中提供的描述符，描述符是可以嵌套使用的。

最后，nnet3 通过以上语法规则将不同的组件拼装起来并初始化，在设计网络结构时，可以通过 `nnet3-init` 校验配置文件能否初始化成功。

语音识别实际问题

7	说话人自适应	47
7.1	什么是说话人自适应	
7.2	特征域自适应与声道长度规整	
7.3	声学模型自适应：HMM-GMM系统	
7.4	声学模型自适应：DNN系统	
7.5	领域自适应	
7.6	小结	
8	噪声对抗与环境鲁棒性	57
8.1	环境鲁棒性简介	
8.2	前端信号处理方法	
8.3	后端模型增强方法	
8.4	小结	
9	新词处理与领域泛化	69
10	小语种识别	71
10.1	写在前面的话	
10.2	小语种及其所面临的困境	
10.3	小语种中的基础数据	
10.4	对小语种声学模型的探索	
10.5	对小语种语言模型的探索	
10.6	小结	
11	关键词唤醒与嵌入式系统	77
11.1	什么是关键词唤醒	
11.2	关键词唤醒和LVCSR	
11.3	关键词唤醒的难点	
11.4	模型方法	
11.5	关键词唤醒和嵌入式系统	
11.6	小结	

7. 说话人自适应

by 启明¹

7.1 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

¹wangd.csit.org

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[7]。

7.2 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）[8]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7.1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[9]。

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[10]。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。

7.3 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7.4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。

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，

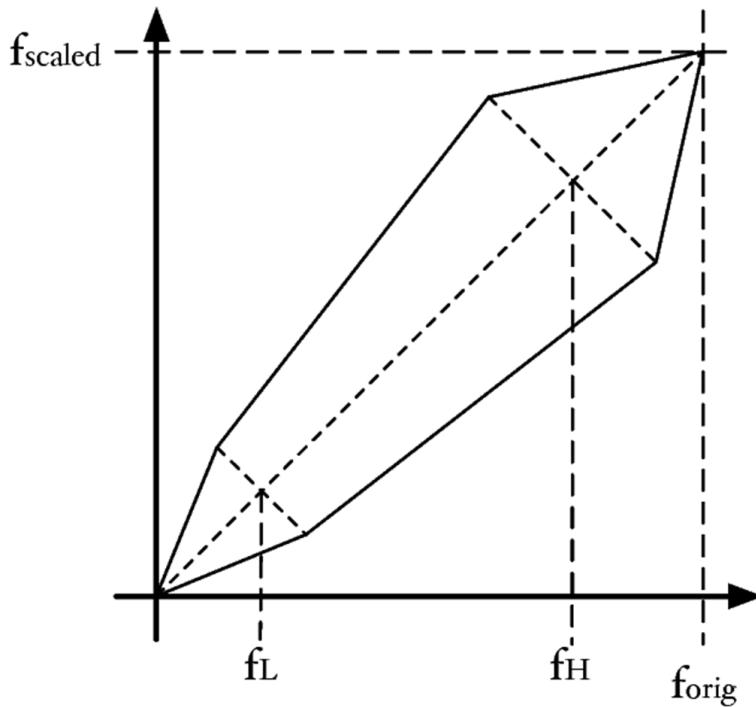


图 7.1: 分段线性VTLN函数。横轴为原始频率，纵轴为变换后的频率。中间虚线代表参考声道长度($\alpha=1$)下的映射，上下两条实线分别代表不同形变因子对应的映射函数。。

```
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)
```

图 7.2: Kaldi中src/feat/mel-computations.cc中实现的VTLN代码。

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

```

# 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

```

图 7.3: Kaldi中wsj recipe下的VTLN步骤。

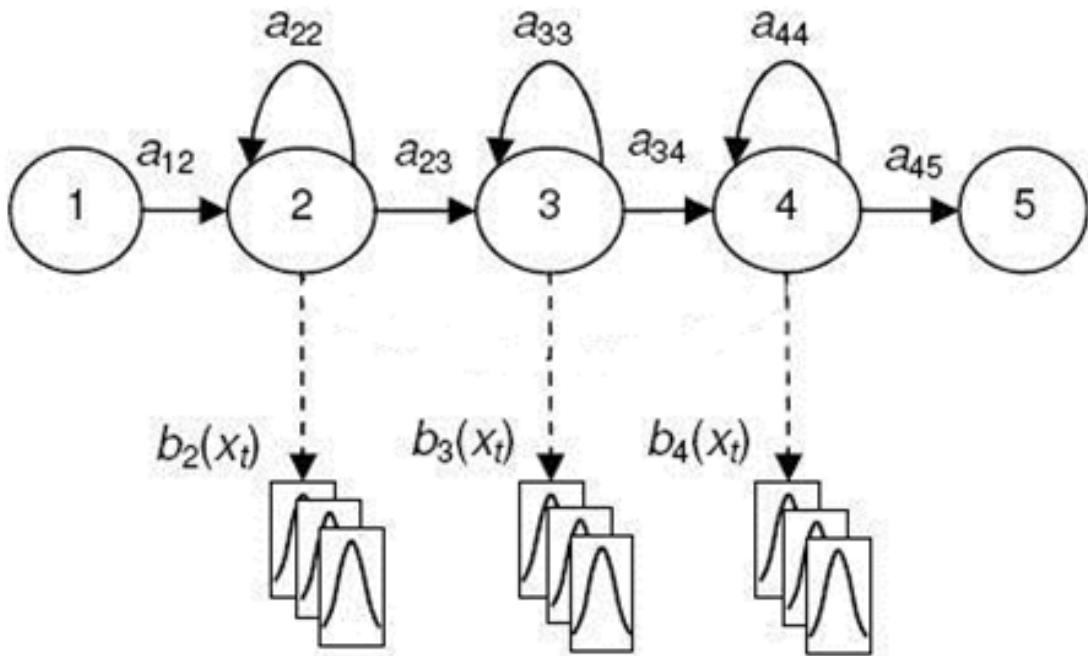


图 7.4: HMM-GMM模型，其中HMM包括三个输出状态，每个状态用一个GMM表示。

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）[11]和最大似然线性回归（Maximum Likelihood Linear Regression）[12, 13]。

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。

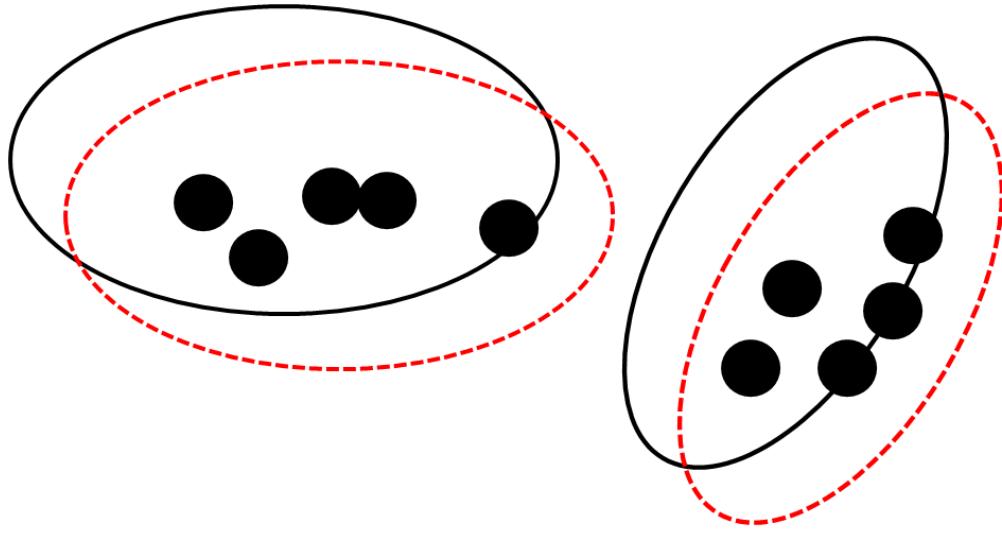


图 7.5: GMM模型的自适应。黑圈表示原始GMM的两个高斯成分，实心黑点表示目标说话人的语音特征向量。可见，这两个高斯成分无法有效描述目标说话人语音。通过对模型进行自适应，两个高斯成分发生了变化，如图中红色虚线所示。

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；对该高斯成分的语音帧为 $X_{ijk} = \{x_{ijk}(1), x_{ijk}(2), \dots, x_{ijk}(T_{ijk})\}$ 。依最大似然（Maximum Likelihood）准则，可计算出该高斯成分的均值如下：

$$\mu_{ijk} = \frac{1}{T_{ijk}} \sum_{t=1}^{T_{ijk}} x_{ijk}(t). \quad (7.1)$$

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：

$$\mu'_{ijk} = (1 - \alpha)\mu_{ijk}^0 + \alpha\mu_{ijk}. \quad (7.2)$$

其中 μ_{ijk}^0 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_{ijk}^0 为均值，以 $\hat{\Sigma}_{ijk}$ 为协方差矩阵的高斯分布。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：

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

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：

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

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(7.2)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7.2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 $strain; then
        steps/align_si.sh --boost-silence 1.25 --nj 10 --cmd "$strain_cmd" \
            data/train_si84_half data/lang_nosp exp/mono0a exp/mono0a_ali || exit 1;
        steps/train_deltas.sh --boost-silence 1.25 --cmd "$strain_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;
        done
    fi
fi
```

图 7.6: Kaldi wsj recipe中提供的alignment过程。

7.3.2 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：

$$L(M) = \prod_{ijk} p(x_{ijk}|M\mu_{ijk}, \sigma_{ijk}). \quad (7.3)$$

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（7.3）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（7.3）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：

$$x'_{ijk} = Mx_{ijk}\Theta$$

这种在特征上进行线性变换的方法称为fMLLR。

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7.7所示），称为SAT训练（Speaker Adaptive Training, SAT)[14]。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。

7.4 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。

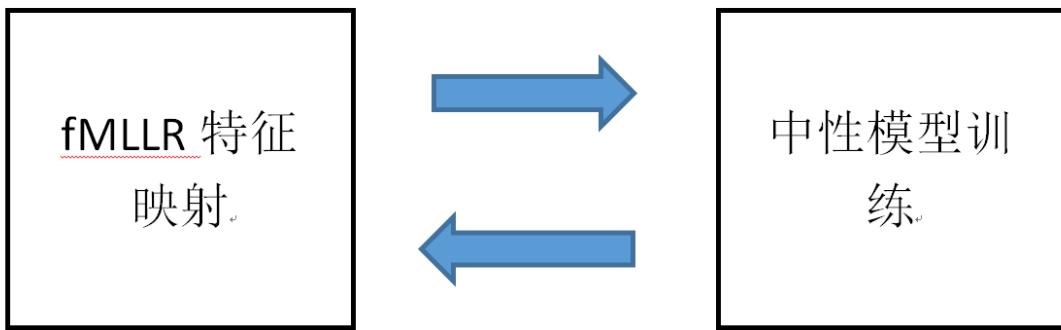


图 7.7: SAT训练。基于fMLLR的特征映射和基于映射后特征的模型训练迭代进行。

```

# local/run_deltas.sh trains a delta+delta-delta system. It's not really recommended or
# necessary, but it does contain a demonstration of the decode_fromlatents.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
fi
  
```

图 7.8: 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。

7.4.1 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[15]。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[16]。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[17]。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[18]。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。

7.4.2 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7.9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[19]。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??所示。

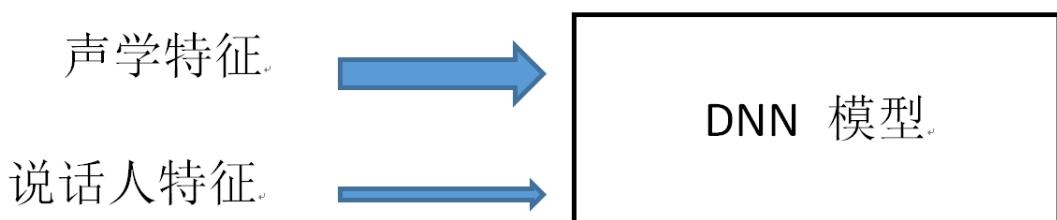


图 7.9: 基于i-vector的DNN条件学习方法。

7.5 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

```

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
done

```

图 7.10: Kaldi wsj recipe 中, nnet3/run_tdnn.sh 中基于 i-vector 的条件学习代码。

用新旧模型插值法实现语言模型自适应[20]。对神经网络语言模型的自适应方法研究的还比较少[21]。

7.6 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。

8. 噪声对抗与环境鲁棒性

by 启明¹

8.1 环境鲁棒性简介

我们常有这样的体验，本来好好的语音输入法，在办公室里基本不会有什么错误，但在大街上使用时就会感到性能明显下降；挺好的语音助手，平常百试不爽，但在公交车上问个问题经常答非所问。这主要是因为实际应用场景中的声学环境非常复杂，这些复杂场景不可能在训练时被全部覆盖，因此形成识别场景与模型的不匹配，导致系统性能的急剧下降。

总体而言，声学场景的复杂性主要可归结为三类：

1. **背景噪音**。实际应用场景中可能包含各种不同类型的噪音，如机器声、汽车引擎声、开门声、背景音乐声、其它人的谈话声等。这些噪音的混入会使语音信号发生显著变化，引起识别性能的下降。
2. **混响和回声**。在一个房间里，发音会在房间四壁反射，形成混响。房间越大，反射回来的声音相对原始声音延迟越久，产生的混响效果越明显。同样的混响效果也发生在电话通信中，在这一场景下，虽然没有象墙壁那样的直接反射，但一方的语音有可能通过对方的听筒和麦克风反射回说话者，形成混响。这种混响一般称为回声。混响和回声会显著降低声音的清晰度，严重影响识别性能。
3. **信道差异**。不同麦克风的物理特性可能有显著差别。如电容式麦克通过电容的充放电产生的电压变化代表声音，而动圈式麦克通过线圈在磁场中运动时产生的电流来代表声音信号。因此，对同一个声音信号，不同麦克风录制的声音会有显著区别。即使同一种麦克风，因信号处理方式的不同，也会得到不同的声音采样。这些差异

¹wangd.csit.org

包括增益设置、静音门限、频域补偿、编解码方式、压缩算法等。我们将上述录音设备、传输媒介等因素的影响称为信道差异。信道差异极大增加了语音识别系统的复杂性，当训练和识别的信道差异较大时，识别系统的性能将明显下降。

一个语音识别系统如果可以对抗实际应用场景的复杂性，在复杂场景下依然可以得到较好的识别性能，我们称之为一个环境鲁棒的识别系统。为了提高识别系统的鲁棒性，研究者提出了各种方法，这些方法大体上可以分为两类：前端信号处理和后端模型增强。前端信号处理方法通过各种信号处理算法减小噪音、回声和信道对语音信号的影响，使之接近正常安静的语音；后端模型增强方法通过对模型做适当调整，使之更加适应实际场景的声学特性。一般来说，前端处理方法计算量小，灵活方便，但性能提升有限；后端模型增强方法计算量较大，需要的数据较多，但性能更好。下面我们将对这两种方法做简单介绍。

8.2 前端信号处理方法

前端信号处理方法通过对语音信号进行一系列变换，以去除信号中各种噪声和失真，恢复原始清晰语音。不同处理算法基于不同假设，产生的效果也不尽相同。总体来说，我们可以将环境影响分为加性噪声和卷积噪声两类，其中背景噪声可以认为是加性噪声，是在原有声音信号上叠加另一种信号，从而产生破坏；混响、回声和信道差异可以认为是一种卷积噪声，是在原有声音信号上的一种附加变换。我们将介绍对不同类噪声的不同处理算法。

8.2.1 语音增强方法

语音增强是一种时域或频谱域上的信号处理方法。历史上，语音增强的目的是为了提高语音相对人耳的可懂度，而不是语音识别性能的提高。尽管如此，在很多情况下，这种方法对语音识别依然有所帮助。

谱减法与加性噪声去除

谱减法（Spectral Subtraction, SS）是一种常用的语音增强方法[22]。这一方法假设带噪语音的能量谱是由原始语音的能量谱和噪音的能量谱简单相加得到的，因此，如果可以估计出噪音的能量谱，即可将该能量谱从带噪语音中减去，从而恢复原始干净语音的能量谱。写成公式为：

$$|\hat{X}(f)|^2 = |Y(f)|^2 - |\hat{N}(f)|^2,$$

其中 $Y(f)$ 和 $\hat{N}(f)$ 分别为带噪语音和噪音的频谱， $\hat{X}(f)$ 为利用谱减法估计出的原始语音频谱。事实上，上述能量叠加假设忽略了原始语音与噪音之间的相关性，因此只能是一个近似估计。谱减法需要估计噪音信号的频谱，这可以通过确定一些非语音帧（如句子的开始片段和结束片段），并对这些帧的能量谱进行平均。然而，基于这一平均能量得到的噪音估计未必能保证每一帧信号做谱减后都是正数，因此需要在应用时做适当调整。这些

调整有可能会导致相邻频谱间变化过于剧烈，从而引入音乐噪音，需要做进一步平滑处理[23]。

回声消除

谱减法处理的是加性噪声。对于回声和混响这种卷积噪声，直接应用谱减法并不合适。为了减小回声和混响的影响，可能估计原始语音到接收端的传递函数，这一函数可用房间脉冲响应（RIR）来表示。基于这一脉冲响应，可以设计一个逆滤波器，使之与RIR的作用互相抵消，从而减弱回声和混响的影响[24, 25]。然而，估计RIR本身就是很困难的问题，而不精确的RIR会严重影响去回声的效果，还可能引入新的畸变。一些研究者发现带混响的语音在做LPC估计时，其残差往往具有更显著的高斯性。基于这一发现，可以直接设计逆滤波器，使生成语音的LPC残差更加非高斯化[26]。这种方法不需估计房间的RIR，防止了错误累积。另外一些研究者关注对高延迟混响的去除，这些混响对语音识别的影响最大。例如在大礼堂中，混响会持续很久，这些持久混响使得频谱结构发生显著改变。研究者设计了一种基于混响时间来估计RIR的方法。一般常用 T_{60} 作为混响时间。所谓 T_{60} ，是语音信号衰减60dB所需要的时间。基于 T_{60} 设计一个RIR模型，从而可以估计出延迟较大的混响，最后利用谱减法将这些混响的能量从带噪语音中去除。另一些研究者利用线性预测模型从历史观察信号中恢复出当前原始信号（注意当前观察信号是由历史信号经过延迟衰减并与当前原始信号叠加而成），从而将逆滤波器的设计问题转化为线性预测模型的参数估计问题[27]。上述这些方法都利用了混响和回声产生的物理机理，因而针对性较强。

麦克风阵列

前面所述的各种去噪和去回声方法都是基于单一麦克风，能利用的信息有限，面对复杂场景时很难得到较好的归一化效果。近年来，多麦克风设备开始普及（例如在几乎所有手机上，都装有两个以上的麦克风），使得我们得以利用语音信号传递过程中的更多空间和时间信息，从而极大提高了语音增强能力。最简单的方法是利用一个远端麦克风录制背景噪音，一个近端麦克风录制说话者语音，通过简单谱减法实现语音增强。更通用的解决方案是麦克风阵列技术[27, 28, 29]。

麦克风阵列（Microphone Array）是按一定几何结构组合在一起的一组麦克风。最常用的阵列包括线性阵列和环形阵列，如图8.1所示。一般来说，阵列中的麦克风都是全指向的，即对各个方向的敏感度是一致的，但当这些全指向麦克风组合在一起的时候，就产生了强烈的指向性，从而实现方向选择、去噪、去混响等强大的功能。我们以一个线性阵列为例来对此进行说明。

考虑如图8.2所示的四麦阵列，每两个麦克风之间的间隔为 l ，阵列的输出为四个麦克风输出的简单加和。对一个频率为 f ，由夹角 θ 入射的平面波，可以计算相临两个麦克风之间的信号延迟为：

$$\Delta t = \frac{l \sin(\theta)}{c},$$

其中 c 为声速。由此可计算相临麦克风之间的相位差为 $2\pi f \Delta t$ 。如果我们将最左边的麦克风接收到的信号记为 $A e^{j 2\pi f t}$ ，则第 i 个麦克风的信号则为： $A e^{j 2\pi f (t + i \Delta t)}$ 。由此，可计算这四路麦克风输出的结果为：

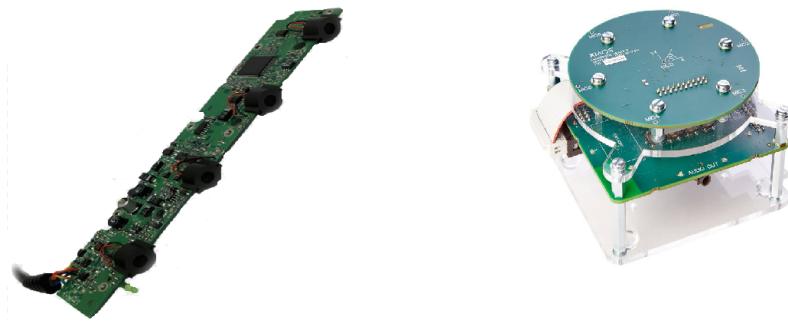


图 8.1: 线性和环形麦克风阵列。

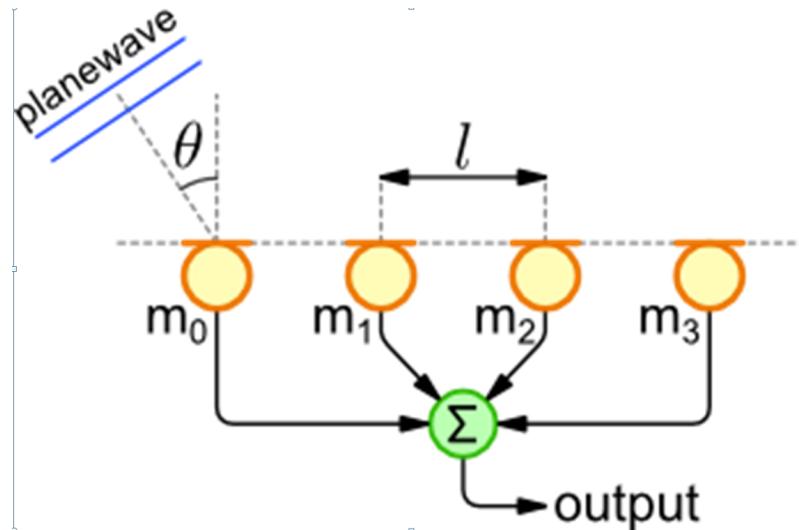


图 8.2: 线性麦克风阵列的声音采集和加和。

$$\frac{1}{4} \sum_{i=0}^3 A e^{j2\pi f(t+i\Delta t)} = \frac{1}{4} \sum_{i=0}^3 A e^{j2\pi f(t+\frac{ilsin(\theta)}{c})}.$$

与单独一个麦克风相比，可知其输出的增益（以dB为单位）为：

$$20\log_{10} \frac{1}{4} \sum_{i=0}^3 e^{j2\pi f \frac{ilsin(\theta)}{c}}$$

由上式可知，该增益是入射角 θ 的函数，如图8.3所示。为了更清晰地对比，图中同时给出了单一麦克风在不同方向上的增益函数。可见，阵列具有明显的方向选择性，只有在正前方的输入才有较好的增益，其它方向的输入都被抑制。这种指向性使得阵列可以选择特定方向的声音，抑制其它方向的声音，从而极大提高信噪比。同时，不同麦克风所接收的噪音是不相关的，这些不相关噪音在互相叠加时会互相抵消，因此可显著消除加性噪声的影响。

对上述简单加和的阵列而言，其增益的指向性是固定的，即只有在正前面的增益最大。如果我们对每路麦克风的输出做适当延迟，再对延迟后的信号做加和，即可选择阵列

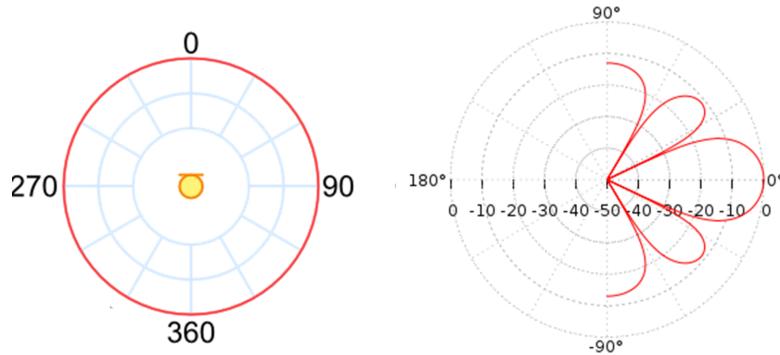


图 8.3: 单一麦克风 (左) 和线性麦克风阵列 (右) 在各方向上的增益。可见，麦克阵列具有明显的指向性。

的指向性。事实上，如果我们想对入射角为 θ 的方向做最大增益，只需对由该入射角引起的延迟 Δt 进行补偿即可，这一方法称为延迟-加和算法 (Delay-Sum Algorithm)，如图8.4所示。为提高延迟-加和算法的性能，研究者提出了各种改进方法，包括为每个麦克风引入增益参数，调节这些参数使之更适合语音识别任务等[30]。

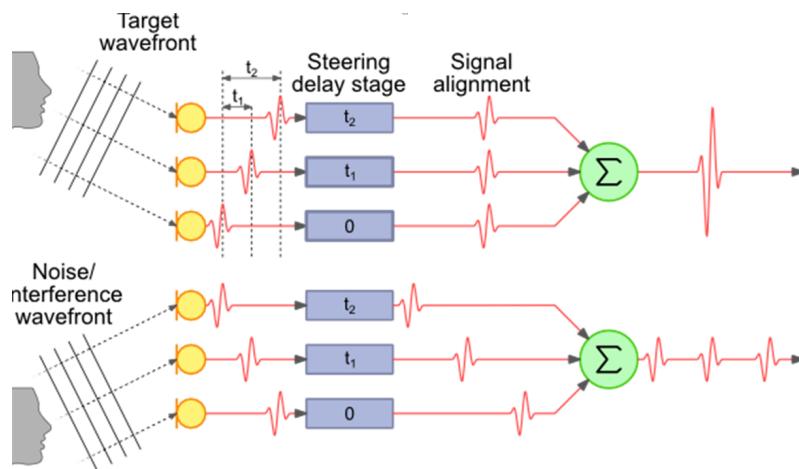


图 8.4: 线性麦克风阵列的延迟-加和算法。对目标语音方向，选择合理的延迟补偿，使得各路麦克风在做完补偿后的相位恰好一致，即可实现该方向的增益最大化。在这一设置下，其它方向的声音因相位失配，在输出信号中的增益减小。

8.2.2 特征域补偿方法

如前所述，语音增强方法的目的是增加语音的清晰度和可懂度，这一目标与语音识别有一定差距。对语音识别系统来说，特征本身的鲁棒性，或环境不变性更加重要。因此，在特征域上的补偿或正规化通常可起到更好的效果。我们将讨论三种方法：倒谱特征归一化 (CMN & CVN)、向量泰勒展开 (Vector Taylor Series, VTS) 和 SPLICE。

CMN & CVN

CMN和CVN是最常用的特征补偿方法，主要用来对卷积噪声进行消除。我们首先讨论CMN。我们已经知道，Fbank和MFCC是最常用的两种特征。这两种特征基于共同的前端处理：加窗、预加重、FFT变换、Mel频谱归整、频域能量加窗、log压缩。对一个在时域上卷积的信道噪声，通过上述过程可以实现分解。为清楚看到这一点，设原始信号为 $x(t)$ ，带噪语音信号为 $y(t)$ ，信道的卷积噪声为 $h(t)$ ，则有：

$$y(t) = x(t) * h(t) \quad (8.1)$$

$$Y(w) = X(w)H(w) \quad (8.2)$$

$$\log|Y(w)|^2 = \log|X(w)|^2 + \log|H(w)|^2 \quad (8.3)$$

由此可得：

$$\mathbf{y} = \mathbf{x} + \mathbf{h},$$

其中 \mathbf{x} 和 \mathbf{y} 分别为原始语音信号和带噪语音信号的Fbank特征， \mathbf{h} 是和信道相关的卷积噪声。因此，如果我们可以估计出 \mathbf{h} ，即可估计出原始语音信号的Fbank特征 \mathbf{x} 。在实际操作中，可以选择信号中的非语音片段来估计 \mathbf{h} 。进一步，如果我们假设 \mathbf{x} 是高斯的，也可以通过对整句语音信号取平均来得到，即：

$$\mathbf{h} = \boldsymbol{\mu}_{\mathbf{y}},$$

$$\hat{\mathbf{x}} = \mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}}.$$

MFCC特征是在Fbank特征基础上加入一个DCT变换，因为该变换是线性的，因此上述分解关系依然成立，即：

$$C\mathbf{y} = C\mathbf{x} + C\mathbf{h}$$

其中 C 是DCT的变换矩阵。由于MFCC是倒谱系数，上述方法称为倒谱均值正规化（Cepstra Mean Normalization, CMN）[31]。

形式上，CMN可以认为是对特征进行一阶归一化的方法。基于这一思路，可以设计一种二阶归一化方法，即对方差进行归一化，称为倒谱方差正规化（CVN）。实际应用中，CVN一般和CMN联合使用，称为CMVN，计算公式为：

$$\hat{\mathbf{x}} = \frac{\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}}}{\sigma_{\mathbf{y}}},$$

其中的除法为按位除。和CMN不同，CVN并没有特别明确的物理背景，但在实际应用中通常会取得一定的性能提高。

CMN和CVN只对一阶和二阶量进行正规化，类似的思路可以扩展到对特征向量的分布进行正规化。一种方法是将特征向量的每一维都正规化到标准高斯分布，称为特征的高

斯化[32]。高斯化通常采用统计方法，以直方图形式统计特征的实际分布，将其变换为累积概率分布，再将该分布映射到标准高斯分布的累积分布。高斯化对一些任务有一定效果，但在某些任务上的表现未必好于简单的CMN。

在实际系统中，为了保证实时性，需要设计一种在线CMN。在对一句话进行识别时，最初没有任何数据，这时采用缺省的CMN参数来对特征进行归一化；当数据逐渐积累后，对CMN参数逐渐求精，从而得到更好的归一化特征。这种在线估计可以理解为是一种高通滤波器（滤掉了固定不变的成份）。将正规化过程表述为一种滤波过程具有很大启发性，一些著名的去噪方法，如ARMA滤波和RASTA滤波[33]都遵循这一思路。

向量泰勒展开（VTS）

向量泰勒展开（VTS）是一种对加性噪声的建模方法。如在谱减法中所述，对于加性噪声，我们假设带噪语音的能量是原始语音和噪声语音的能量之和。假设我们使用的是Fbank特征，这一关系可表示为：

$$e^{\mathbf{y}} = e^{\mathbf{x}} + e^{\mathbf{n}}.$$

做简单变换，有：

$$e^{\mathbf{y}} = e^{\mathbf{x}}(1 + e^{\mathbf{n}-\mathbf{x}}),$$

$$\mathbf{y} = \mathbf{x} + \ln(1 + e^{\mathbf{n}-\mathbf{x}}).$$

如果记 $\mathbf{r} = \mathbf{n} - \mathbf{x}$ ，且：

$$g(\mathbf{r}) = \ln(1 + e^{\mathbf{r}}).$$

可得如下关系：

$$\mathbf{y} = \mathbf{x} + g(\mathbf{r}).$$

注意， $g(\mathbf{r})$ 是一个非线性函数。如果对这一非线性函数做一阶泰勒展开，即可得到带噪语音、原始语音和噪音之间的简单对应关系，从而由带噪语音推导出原始语音，这一方法称为VTS方法。一般假设原始语音具有混合高斯形式，噪声具有高斯形式。在这一假设下，可通过迭代求出在每一个高斯成分 s 下， \mathbf{r} 的期望 $\mu_s^{\mathbf{r}}$ ，并由此得到基于状态 s 的原始语音估计 $\hat{\mathbf{x}}$ 如下[34]：

$$\hat{\mathbf{x}} = \mathbf{y} - \ln(e^{\mu_s^{\mathbf{r}}} + 1) + \mu_s^{\mathbf{r}} \Theta$$

由上述推导过程可知，VTS的基本假设是噪声是加性的，因此语音和噪声之间的能量具有加和关系。基于这一基本假设，VTS推导出基于Fbank特征，带噪语音和原始语音之间的关系，并用泰勒展开对这一关系进行近似。值得说明的是，对倒谱特征，如MFCC，上述推导过程依然成立，只不过需要加入一个DCT变换。

SPLICE

SPLICE是另一种对特征进行建模的方法。和VTS不同，SPLICE并不假设噪音的加性，而是直接对原始语音和带噪语音的特征向量建立联合概率分布。为保证建模精确性，SPLICE采用GMM模型：

$$p(\mathbf{y}, \mathbf{x}) = \sum_{k=1}^K p(\mathbf{x}|\mathbf{y}, k)p(\mathbf{y}, k)$$

其中 $p(\mathbf{y}, k)$ 也是一个GMM：

$$p(\mathbf{y}, k) = p(\mathbf{y}|k)p(k).$$

SPLICE定义条件概率 $p(\mathbf{x}|\mathbf{y}, k)$ 具有如下线性形式：

$$p(\mathbf{x}|\mathbf{y}, k) = N(\mathbf{x}; A_k \mathbf{y} + b_k, \Sigma_k),$$

则可由带噪语音估计出原始语音：

$$\hat{\mathbf{x}} = \sum_{k=1}^K (A_k \mathbf{y} + b_k) p(k|\mathbf{y}).$$

SPLICE模型中的 $p(\mathbf{y}, k)$ 部分可通过对带噪语音的GMM建模实现，而条件概率 $p(\mathbf{x}|\mathbf{y}, k)$ 中的参数 $\{A_k, b_k\}$ 一般需要基于原始语音和相应的带噪语音数据（Stereo Data）进行训练。

8.2.3 基于DNN的特征映射

前面所述的大部分方法都假设了一个物理过程，基于该物理过程进行建模。这些方法具有较好的理论基础，需要的数据和计算量通常较小。然而，这些建模方法都或多或少引入了一些人为假设，这些假设在实际应用中可能无法满足，带来模型的不精确性。同时，对一些难以建模的场景（如传输过程中信道的即时改变），这些方法也很难凑效。近年来，深度神经网络（DNN）成为语音信号处理的强大工具。DNN的一个显著优势是可以近似任何映射函数，因此可以学习任何复杂的信号传递过程。我们可以利用这一能力，基于DNN将复杂环境中的语音信号或特征映射成安静环境下的信号或特征。研究表明，基于DNN的特征映射方法可取得非常好的效果[35, 36, 37]。

去噪自编码器（Denoising Auto Encoder, DAE）是一种常见的特征映射模型。和SPLICE一样，我们需要准备一份干净数据和一份相应的带噪数据，将带噪数据输入DAE，输出的目标是对应的干净数据。通过训练DAE的参数，即可学习到由带噪语音（或特征）还原出原始语音（或特征）的映射函数。图8.5给出一个利用DAE去除音乐噪音的例子[38]，可以看到，DAE可以有效恢复被音乐破坏的语音数据。

Kaldi的THCHS30 recipe提供了DAE训练流程样例，如图8.6所示。在这一流程中，对每一条训练语句，随机选出一种噪声（基于狄利克雷分布）以及噪声大小（基于高斯分布），将该噪声按选定的大小混入训练语句中，形成DAE的一个训练对。在训练时，对原始数据和加噪后的数据分别提取Fbank特征，DAE学习由带噪特征到原始特征的映射函数。

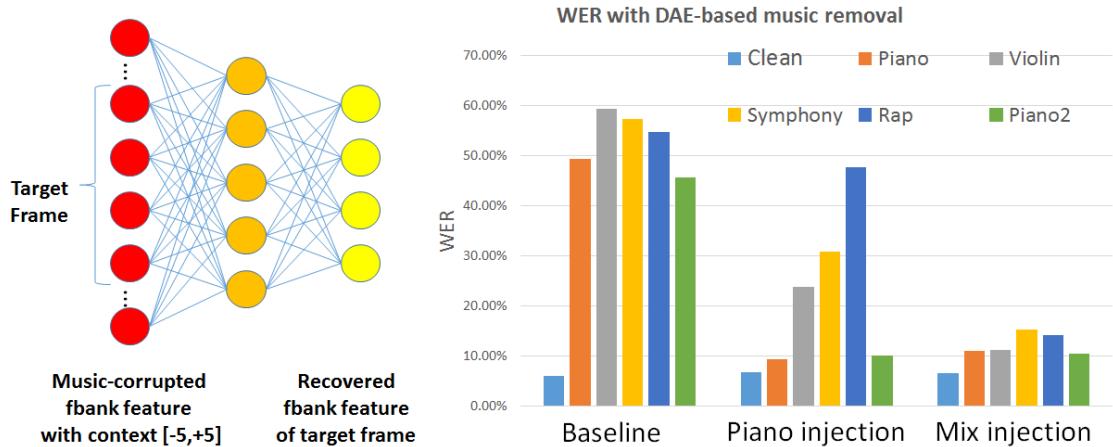


图 8.5: 基于DAE的音乐噪音消除。左图是DAE的结构，右图是实验结果。其中每一组直方图表示一个系统，每一组里的一个直方图代表测试数据中包括某种音乐的结果。从第一组结果可以看到，基于原始模型，在测试数据中加入音乐后性能显著下降；从第二组结果可以看到，即使只加入一种音乐做DAE训练，也可明显提高系统性能，即使对没见过的音乐也是如此；从第三组结果看到，当加入更多类型的音乐进行训练后，性能有了进一步提高。

8.3 后端模型增强方法

后端模型增强方法通过调整声学模型，使系统适应实际应用场景。依系统模型种类不同（如HMM-GMM 或DNN），模型增强的方式也不同。我们主要讨论三种模型增强方法：基于噪声模型的简单模型增强方法、模型自适应和带噪训练。

8.3.1 简单模型增强

对于一个HMM-GMM系统，如果我们假设噪音是加性的，则在8.2.2一节所讨论的对特征的补偿方法可以同样用于对模型参数的补偿。和特征补偿相比，这种在模型上的补偿更加灵活，性能通常也更好。

以Fbank特征为例，HMM-GMM系统假设分配到每个高斯成份上的语音帧（Fbank）是高斯分布的，同样，噪声也是高斯分布的。这一高斯分布映射到频域能量谱上，分别记为 X 和 N ，这两者也是随机变量，分布为对数高斯（即取对数后是高斯的）。引入加性噪声假设，得到带噪语音的能量谱为 $Y = X + N$ 。一般来说，视 X 和 N 的相关性以及它们的相对强弱， Y 的分布是不规则的。如果我们假设 Y 依然对数高斯的，即可求出对应到Fbank域的高斯分布的参数，由此实现对原模型的修正。这一方法称为平行模型加和（Parallel Model Combination）[39]。

另一种方法是将特征域上的VTS补偿应用到模型增强，即不对特征进行修正，而是对模型参数进行改进，以更好描述加噪后的语音。这一方法同样用到加性噪声假设，但和PMC中的对数高斯近似不同，VTS基于泰勒展开对 y 和 x 之间的关系做近似[40]。

上述两种方法相对简单，但只能处理加性噪声，且只能应用于HMM-GMM系统，现

```
#quick ali
steps/align_fmllr.sh --nj $n --cmd "$strain_cmd" data/mfcc/train data/lang exp/tri4b exp/tri4b_ali || exit 1;

#quick ali_cv
steps/align_fmllr.sh --nj $n --cmd "$strain_cmd" data/mfcc/dev data/lang exp/tri4b exp/tri4b_ali_cv || exit 1;

#train dnn model
local/nnet/run_dnn.sh --stage 0 --nj $n exp/tri4b exp/tri4b_ali exp/tri4b_ali_cv || exit 1;

#train dae model
#python2.6 or above is required for noisy data generation.
#To speed up the process, pyximport for python is recommended.
local/dae/run_dae.sh $thchs || exit 1;
~
~
~
```

```
#!/bin/bash
# Copyright 2016 Tsinghua University (Author: Dong Wang, Xuewei Zhang). Apache 2.0.
#           2016 LeSpeech (Author: Xingyu Na). Apache 2.0

# Conducts experiments of dae-based denoising

stage=-1
nj=8

. ./cmd.sh ## You'll want to change cmd.sh to something that will work on your system.
            ## This relates to the queue.

. ./path.sh ## Source the tools/utils (import the queue.pl)
. utils/parse_options.sh || exit 1;

thchs=$1

if [ $stage -le -1 ]; then
    echo "DAE: switching to per-utterance CMVN mode"
    for x in train dev test test_phone; do
        mv data/fbank/$x/cmvn.scp data/fbank/$x/cmvn.scp.per_spk
        mv data/fbank/$x/spk2utt data/fbank/$x/spk2utt.per_spk
        mv data/fbank/$x/utt2spk data/fbank/$x/utt2spk.per_spk
        awk '{print $1 " " $1}' data/fbank/$x/utt2spk.per_spk > data/fbank/$x/utt2spk
```

图 8.6: Kaldi THCHS30 recipe 中提供的 DAE 训练和识别脚本。上图是 run.sh 中的调用程序，下图是 local/dae/run_dae.sh 脚本。

在已经较少应用。

8.3.2 模型自适应

如果我们将实际应用环境看作是与训练不同的另一种声学场景，则可基于“说话人自适应”一节中所提到的领域自适应方法，利用应用场景的数据对模型进行更新。对HMM-GMM系统，一般采用MAP和MLLR两种方法；对DNN系统，可在原模型基础上进行再训练，训练时选择较小的步长；或采用知识迁移方法[41]，以原系统的输出作为约束，以减小过拟合的风险。当前对DNN模型最有效的自适应方法还是基于i-vector的条件学习方法。前面提到过，i-vector事实上是一种全信息向量，包括说话人、信道、语言、情绪等多种长时信息，因而可以充分覆盖噪声、混响、编码方式等环境因子的变化。实验表明，将i-vector作为一种辅助信息引入到DNN模型训练和识别过程中，可以非常有效地对抗环境影响。

8.3.3 多场景学习和数据增强

DNN的一个显著优势是可以进行多场景进行学习。在传统GMM-HMM系统中，虽然我们可以通过收集更多实际应用场景的数据来提高系统性能，但由于模型限制，当收集的数据具有较大差异性时，将导致音素的区分性下降。这意味着大量数据虽然可以提高对

场景的覆盖能力，但对某一应用场景来说，多场景学习并不能达到单一场景建模的效果。DNN极大改变了这种状况。实验表明，DNN模型可以有效学习多场景下的数据，这些各异场景的数据不仅不会降低音素的区分性，反而会互相促进，在各种场景下都能得到同步提高[42]。这一结果具有重要意义，说明如果我们可以收集到足够多、对场景覆盖足够全的数据，那么一个DNN系统即可在所有场景下顺利工作。这事实上已经在原则上解决了环境鲁棒性的问题。某种程度上说，DNN的这种多场景学习能力是今天大规模商用语音识别系统的基础。

尽管如此，我们依然要考虑如何有效利用DNN的这种多场景学习能力。这是因为数据天然具有长尾效应：绝大部分数据可能是正常的，但对很多特别场景（如特别强的噪音，特别强的混响，很少用的编码方式等），数据通常是不足的。数据在场景上的分布不均衡事实上带来了另一种更深刻的环境鲁棒性问题。数据增强（Data Augmentation）[43]，或带噪训练（Noisy Training），是解决数据不均衡问题的有效方法。具体来说，数据增强方法对原始训练数据进行各种变换，以模拟不同场景下语音信号的变异情况。这些模拟包括在数据中随机加入不同类型的噪声，让数据通过随机生成的RIR，通过各种编解码器进行重构等。实验发现，数据增强方法可极大提高系统的鲁棒性，特别是提高非典型场景下的识别性能[44, 45]。

8.4 小结

我们简要介绍了提高识别系统鲁棒性的几种方法，这些方法可分为前端信号处理和后端模型增强两类。前端信号处理方法的目的是对不同环境下的语音信号或特征进行归一化，使之可以适应标准语音训练出的模型。最常用的前端处理方法是CMN，这种方法简单、高效，且有明确物理意义，被广泛应用于各种商用识别系统。另一种前端处理方法是基于DAE的特征映射。归因于神经网络强大的函数学习能力，DAE可以实现对各种复杂声学场景的归一化。后端模型增强方法的基本思路是对模型进行改进，使之对目标场景有更好的识别效果。模型增强的主要方法有两种，一是对模型进行自适应，使其适应目标场景，二是多场景学习，提高模型的泛化能力。对DNN来说，多场景训练一般可取得较好的效果，利用数据增强方法可以进一步提高对非典型场景的覆盖。

9. 新词处理与领域泛化

10. 小语种识别

by 石颖¹

10.1 写在前面的话

16年8月份我初到语音和语言技术中心实习，当时的我只笼统的读过李航老师的撰写的《统计学习方法》，脑海中只有肤浅的机器学习的概念，更是不知深度学习为何物。然而，幸运的是，我来到实验室不到一个月的时间，就赶上了人生中的第一个deadline，ICASSP 2017。可以说我对kaldi，乃至语音识别的入门是从这个deadline开始的。Deadline结束后，恰逢中心启动了国家自然科学基金的重点项目M2ASR(Multilingual Minorlingual Automatic Speech Recognition)，于是乎王老师丢给我80小时的哈萨克语的语音数据，让我跑个baseline… 直至今日，该项目的大部分工作都是由汤博士指导，我来执行的。真不知道王老师当时哪里来的勇气，敢于把这样的任务交给我这个还算入门的实习生来做。不过我着实应该衷心的感谢王老师对于我的信任，也许没有当时这个任务，我很可能就流落到某某公司泯然众人了。

写上一段的目的不是为了标榜自己，更不是为了吹捧我们中心，而是想告诉读者，语音识别入门以及kaldi入门，没有想象中的那么困难。甚至于深度学习中的大部分框架及理论都是较为朴素的。当然这里的朴素仅限于入门，如果恰逢某读者胸怀大志，早已将语音识别以及kaldi 的基础知识烂熟于胸，想要在语音识别领域有王老师那样的建树，请略过此章节或者直接私信王老师 wangd99@tsinghua.mails.edu.cn.

¹shiying.cs.lt.org

10.2 小语种及其所面临的困境

回到本章的主题：小语种语音识别。首先介绍什么是小语种。众所周知，我国的官方用语为汉语普通话，然而我们生活在一个由56个民族组成的大家庭，因此我国有着极其丰富的语言体系，小语种与方言错综复杂，甚至于有的时候小语种和方言被混为一谈，实则不然。小语种本质上是一种独立的语言，有独立且完备的发音体系，书写方式及语法现象等。国际公认的对小语种的定义为：小语种是除联合国通用语种：英语，中文，法文，俄语，西班牙语和阿拉伯语以外的所有语种。由此可见，小语种是一门独立的语言，这是小语种与方言最本质的区别。而在具体的语音识别任务中，针对方言语音识别的解决方案也比小语种要简单的多。因为我们可以很容易的得到普通话到方言的映射，因此普通话识别用到的语言模型，词表乃至 phone 表都可以直接被方言复用，在声学模型方面，我们只需要用一部分数据对已有的普通话模型进行 tuning 就可以得到很好的结果。然而同样的方法在小语种上往往是行不通的。

在语音识别领域，小语种面临的两大难题是：资源上的“小”以及形式上的不规范。解决了这两个难题，我们就可以很轻松的构建一套可信的语音识别系统，然而资源的收集与规范化是一项极其费时，费力且费钱且枯燥的工作，因此无论在学术界还是在产业界，语音识别在小语种上的进展都是十分缓慢的。



图 10.1: 中国语种分布图

10.3 小语种中的基础数据

10.3.1 音频数据

上一节中，我们已经提到了小语种语音识别面临的第一个问题是“小”，这主要是因为小语种的适用人群相对较少，且小语种分布的地区信息化尚不发达。导致收集语音数据的成本偏高且质量难以控制。因此合理且充分的利用我们手头仅有的音频数据变得尤为重要，对于语音识别来讲，音频数据首先要满足的条件是尽量涵盖该语种全部的单个音素的

发音。以汉语拼音为例。理论上我们在收集语音数据时，应该使得语音数据尽可能的覆盖汉语拼音所有可能的tri-phone(前文中曾提到过tri-phone的概念)，在无法满足该条件时，我们应该保证音频数据可以覆盖所有的汉语拼音中的声母与韵母。在满足了音频数据对发音的覆盖度之后，我们仍需考虑数据对场景的覆盖度，如：噪声，混响，音量高低语速快慢等。对于场景的问题，我们往往采用数据增强的方式进行弥补，即：人为的往语音数据中添加不同场景的噪音，不同距离的混响，通过信号处理的方式调整音频的音量大小及语速快慢。数据增强往往可以使数据在数量上成倍增加，在质量上也能使数据对场景的覆盖更加全面，经试验证明，该方式可以有效的提高语音识别系统的性能。

10.3.2 文本数据与发音词典

文本数据收集：文本数据相对于语音数据来讲更加容易收集。文本数据在语音识别中主要被用于构建语言模型，因此文本数据同样要求场景的覆盖度。例如：文本语料如果大多是新闻稿等一些较为正式的书面内容，则该系统对于口头用语的识别效果会很差。对于语言模型我们可以以ppl为指标作为度量(srilm中提供了计算ppl的接口)。一个经久不息的爬虫可以帮我们收集大量的文本数据(BTW: 如果发现爬到的文本数据对于口头语ppl较高，可以多爬取一些论坛的评论，毕竟人们所发的牢骚大多来源于生活)。

文本数据处理：小语种文本处理最令人头疼的问题是，技术人员看不懂。使用拉丁文作为载体的语言我们尚可分析一二，然而对于维语、哈萨克语、柯尔克孜语等，以阿拉伯字母作为书写载体的语言往往最令我们左右为难(每个字母看起来都差不多)。所以我们建议读者在处理小语种文本时，将小语种的文本映射到拉丁字母上，便于我们分析结果。如果没有语言学家为我们制定映射表或映射规则，我们可以直接将小语种的中的文字转化为其对应的UTF-8编码，这样不仅保证了唯一性，同时也为结果的分析提供了可能。

发音词典lexicon：发音词典是语音识别的核心组成部分。然而对于极其濒危的语种，不排除没有可靠发音词典的可能。在以往的工作中，我们尝试过使用鲁棒性较强的语音识别模型，如：汉语语音识别模型。对小语种的音频做强制对齐(force alignment)以达到使机器自动生成发音词典的目的。遗憾的是这项工作并没有成功。原因也很容易解释，汉语的发音空间并不能代表人类发音的全空间。即汉语模型达到了汉语识别的局部最优，无法对小语种进行无偏估计。如果我们能够构建人类发音的全空间，即学习到一组参数可以表征人类的全部发音，则发音词典的构建将不再依赖语言学家。在今后的工作中，我们将在此方向上做更多的尝试。

10.4 对小语种声学模型的探索

10.4.1 迁移学习：transfer learning

上一节我们提到了使用数据增强(Data augmentation)的方式人为的增加语音数据的总量以及覆盖度，这种方式可以解决一部分问题，然而，通过数据增强得到的语音信号，与纯正的语音信号还是存在着相当的差距。例如：我们无法使用数据增强的方式，使数据覆盖更多的说话人。数据增强模拟的噪音数据有限且无法覆盖更多的信道。

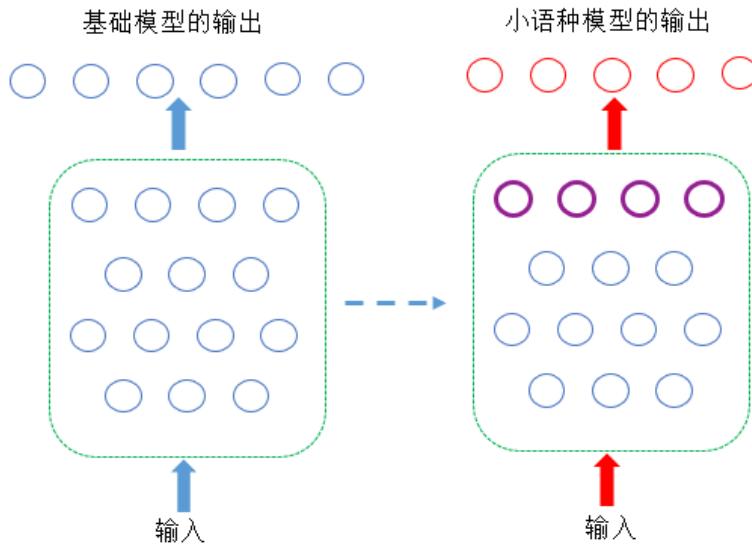


图 10.2: 迁移学习

因此我们想到了迁移学习(Transfer learning)，虽然两种语种的发音空间不完全相同，但是噪声，信道，说话人等无关信息的模式是可以相同的。另外，两种不同的语言的发音空间应该有可共享的部分，例如：汉语拼音中的“i”与英语中的“ɪ”基本无异。而迁移学习的本质恰恰就是让待训练的模型站在“巨人”的肩膀上。具体的做法是如图10.4.1所示，使用将基础模型的输出层，替换为小语种的输出。同时保持基础模型较低的数层不随网络的训练而更新(图中蓝色部分为不更新部分，紫色部分为可更新的节点。具体固定层数由小语种的数据量决定，完全不固定会产生训练数据与模型参数量不匹配的问题。从而导致欠拟合)。进过我们实验的论证，迁移学习对于数据量较少的小语种识别有着很好的效果。

10.4.2 MaR:Map and relabel

迁移学习确实可以在一定程度上解决小语种的问题，然而当我们遇到数据极缺的濒危语种，比如我们只有1小时的标注数据。此时迁移学习也会显得捉襟见肘，1小时的标注数据甚至无法训练出一个较好的高斯混合模型(GMM)，为了应对这样的问题，我们在2018年初，提出了一种新的方法。Map and relabel(MaR)。通过该方法，我们可以使用极少数的带音素级别标注的语音，生成大量的带标注的语音，从而解决数据稀缺的问题。该方法主要分为两步。

Map: 受迁移学习启发，我们可以让模型站在巨人的肩膀上，不同的是，在Map步我们希望模型能够使用极少数数据，学到输入特征到音素的映射。常规的语音识别模型的输出为Ppdf(前文中有相关概念的介绍)。Ppdf是tri-phone的子单元。当训练数据极少时我们无法对数量较为庞大的Ppdf建模，因此，我们可以退而求其次，一个语言的音素数量是远少于Ppdf个数的。我们可以通过迁移学习的方式，使用基础模型对目标的语种的单音素进行建模。如图所示10.4.2。我们替换基础模型最后的线性映射，使模型的输出对应到小语种的单因素，同时固定模型的其他隐藏层，此时整个模型可更新的参数只有最后一层，而将

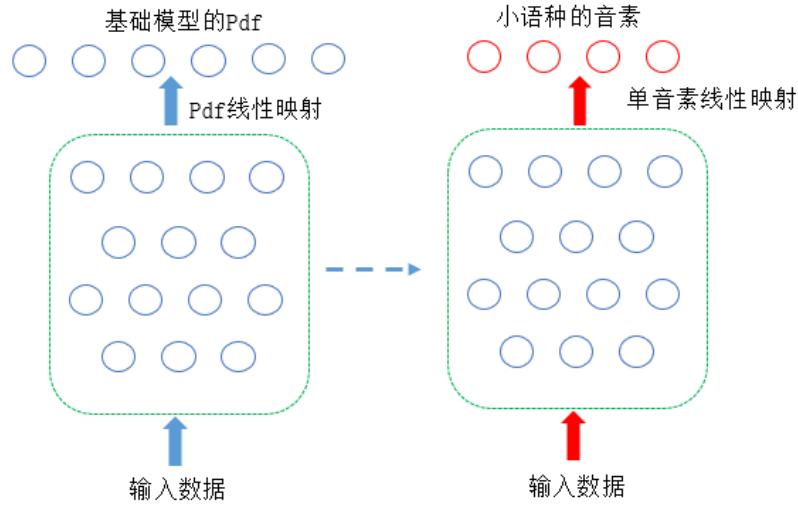


图 10.3: Map

输出改为单音素之后这一层的参数会变得很小。因此我们只需要少量的数据就可以将这一层调整很好。

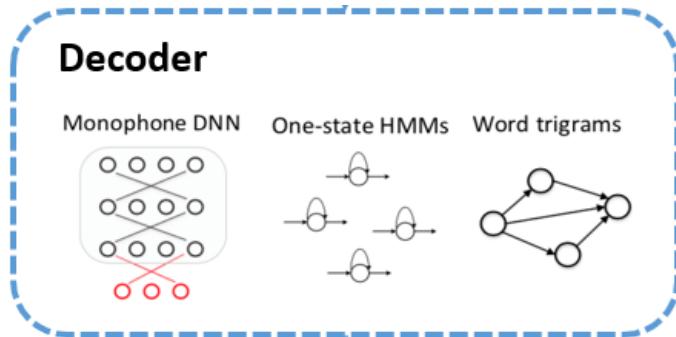


图 10.4: Map

Relabel: Map完成后，我们会得到一个基于迁移学习的分类器，该分类器可以将输入特征分类为不同的单音素，然而使用DNN进行强制对齐并不是一个很好的选择，上文提到过基础模型无法对小语种进行无偏估计。因此Map步得到的分类器，是基于单音素的弱分类器。强制对齐的输出会掺杂大量的毛刺，例如连续的音素串“a a a a”被识别为“a b a c a”。幸运的是一个词中包含了音素的上下文信息，语言模型中又包含了词的上下文信息，因此语音识别的解码器天然具备平滑机制，我们只需要将传统的基于HMM(前文有HMM相关的介绍)三状态的解码器，调整为基于HMM单状态的解码器即可。其中每一个转态对应一个音素。如图10.4.2所示。通过使用解码器对输出进行平滑，我们很容易就可以将一条语音较准确的标注到音素级别。

基于MaR我们可以将大量的无标注数据，标注到音素级别，这种方式可用于极其濒危的语种的拯救。

10.5 对小语种语言模型的探索

在小语种语音识别中，我们采用的语言模型，均为基于统计的n-gram模型，这种模型对于oov(out of vocabulary)词汇完全没有抵抗力，而且这种基于统计的语言模型，在词表较大的时候非常浪费存储与计算资源。所以，我们建议读者多做一些基于RNN的语言模型的尝试，惭愧的是，我们目前并没有开展过多的关于RNN语言模型的研究，在此，我们将基于ngram统计语言模型，向大家分享我们在小语种语言模型上的探索。

在之前的工作中，我们主要面对的是阿尔泰语系的小语种，诸如，维语，哈萨克语，柯尔克孜语等，这些语言有个共同的特点，均属于黏着语。及一个单词是由一个词根加多个词缀组成的(据新疆的小伙伴称，维语有个词是由一个词根加50多个词缀构成的)，这种特性带了两个问题，1.黏着语的词表极其庞大。2.一个较长的词如果开头没有识别出来，那么整个词都无法被识别出来。为了解决这个问题，我们采用了基于词根词缀的解码，将一个单词拆分成一个词根加数个词缀，这种做法的最直接的好处是，大大减小了词表的体积。同时，也可以避免一个单词因为一部分识别错误而导致整个词都被识别错误的情况。(被识别成词根词缀的单词可以通过后处理的方式还原为整个单词)

10.6 小结

本章节中，我们主要向大家阐述了小语种语音识别所面临的困难以及我们为了应对这些困难所做的一些尝试。总体来说可以分为两方面，数据层面，在收集数据整理数据的同时，我们要尽可能的让手中的数据发挥最大的价值，因此数据增强是一个不错的选择。在模型方面，我们始终贯穿一个主线：没有就借。所以有了迁移学习在小语种上的应用以及后来的借助基础模型以MaR的方式生成数据的方法。在今后的工作中，我们可能会更多的投入到对RNN语言模型的探索中去。

目前市场上还没有太多关于小语种语音识别的成熟产品，这并不代表着，小语种语音识别没有应用前景与市场价值，我国少数民族人口众多，仅维语就有1300万使用人群，所以小语种的应用前景还是非常广阔的，然而，由于小语种数据稀缺，变种复杂且缺乏规范性，因此，无论是学术还是产业在小语种上的进展都十分缓慢，所以我们呼吁那些拥有小语种数据的研究机构或者公司能更我们一起进行数据共享。共同推动小语种语音识别的发展。

11. 关键词唤醒与嵌入式系统

by 嘉瑶

如今在科幻电影或小说中，人们对未来世界的想象都是人和机器可以像人和人一样顺畅地交流，其实现在人机语音交互已经逐渐渗透到我们的日常生活中。设想这样的场景，当你满手油污在厨房做饭时，突然忘了一道菜式的做法，你再也不需要手忙脚乱地找手机，你只需要通过说“Hi, Siri”来启动手机，打开APP里的菜谱；当你在寒冬的周末享受温暖的被窝时，突然想享受余音绕梁的美妙，但又被寒冷束缚了手脚，别担心，现在你只需要对着语音助手说，给我放一首《快让我在雪地上撒点野》！

看，语音人机交互已经慢慢地改变了我们的生活方式，改变手动操作机器的旧习，让语音打破空间的距离。而这些技术是如何实现的呢？本章就来揭开语音交互第一步的神秘面纱——关键词唤醒。

关键词唤醒是语音交互的第一步，他就像机器的触发开关，而神奇的是这个开关将由人发出的声音来控制。机器不再是乱接话的“讨厌鬼”或是不爱搭理你的“高冷帝”，只要你说出正确的关键词，就会打开机器的话匣子，让机器像一个听话的助手一样，完成你赋予的任务。他让人类和的机器交流将变得更加灵活，更加可爱，更加人性。

11.1 什么是关键词唤醒

11.1.1 关键概念

什么是关键词唤醒，其实上面描述的两个场景已经解释了这个技术。关键词唤醒就是用户说出特定的语音指令——关键词，设备从休眠状态切换到工作状态，给出指定的响应。关键词唤醒被称为keyword spotting(简称KWS)，定义为在连续语音流中自动检测出关键词的任务[1]。应用场景涉及到生活的方方面面：手机的虚拟助手让手机从“手”中解

放；智能家居让家居生活更加便利；车载语音控制让驾驶更加安全；机器人上搭配语音助手也许未来我们真的能收获大白这样暖心的机器人伙伴。

关键词一般是系统设计时给定，比如苹果手机的“hi, siri”；天猫精灵语音助手的“天猫精灵”；亚马逊Echo 的“Alexa”；Google Home的” Hi, Google “等。当产品检测出相应指定的关键词时，该产品才能被唤醒。关键词的设计对系统的性能指标影响也较大。一般关键词设计为3到5个字，4个最佳。可以使用纯英文、纯中文或中英混合的关键词。关键词音节覆盖越多，差异越大，系统的稳定性越好。当然，未来的产品可能会设计成用户自定义关键词，这样对产品自身系统稳定性的要求将更高。

11.1.2 评价指标

评价一个关键词唤醒系统性能的指标主要有以下几个：

唤醒率 唤醒率即在测试过程中系统被正确唤醒的次数除以总的测试次数。被正确唤醒的意思就是当环境给出关键词时，系统响应并启动。显然，唤醒率越高，系统性能越好。

虚警率 虚警率即在测试过程中系统不该被唤醒的次数除以总的测试次数。不该被唤醒即环境未给出关键词但是系统错误地响应。虚警率越低，系统性能越好。

实时率 实时率即系统的反应速度。用户说出关键词，系统应当能快速反应，这样才能提高用户体验。如果关键词唤醒系统的反应速度像《疯狂动物城》里的那只树懒一样，估计没人会喜欢这样的产品吧。

功耗水平 低功耗是系统性能的一个重要指标。因为很多产品是在充电后使用的，因此这个指标和用户体验紧密相关。

11.1.3 方法流程

论文[46]中给出了关键词唤醒的具体方法并从数学角度给出了详细的阐述。可以将关键词唤醒分为两个阶段：第一个阶段是**检测阶段**。即系统收集关键词在给定的句子中的信息。定义 $X \in \chi$ 是输入语音，其中 χ 是所有语音信号的域（比如语音信号的声学特征向量）。为简化问题，我们假设最多只能有一个关键字能出现在句子中。定义 $K \in \kappa$ 是我们需要检索的关键词，其中 κ 是关键词的域（例如关键词的文本格式），**检测阶段**需要学习一个从语音信号空间 $\chi \times \kappa$ 到置信空间 \mathbb{R}^d 的映射，设参数为 $\theta_1 \in \Theta_1$ ：

$$E_K = f_1(X, K; \theta_1) \quad (11.1)$$

其中 $E_k \in \mathbb{R}^d$ 代表关键词 K 在给定语音信号 X 中出现的置信度。常用的向量置信度维度特征有关键词的后验概率 $P(K | X)$ ，关键词的起始时刻 t_{Kb} 和结束时刻 t_{Ke} 。因此

$$E_K = [P(K|X), t_{Kb}, t_{Ke}] \in \mathbb{R}^3 \quad (11.2)$$

其他附加的维度信息也包括关键词的先验概率 $P(K)$ ，关键词的持续时长 T_K 等。第二个阶段是**决策阶段**。即系统根据检测阶段得到的置信向量 E_K 判断关键词 k 是否出现在语音 X 中，并给出关键词出现的位置。定义输出 O_k 为三元组 $O_k = (Yes/No, t_{Kb}, t_{Ke})$ 。**决策阶段**需要学习一个 E_K 到 O_k 的函数映射，设参数为 $\theta_2 \in \Theta_2$ ：

$$O_K = f_2(E_K; \theta_2) \quad (11.3)$$

将上述两个阶段结合，关键词唤醒任务其实就是学习一个复合函数 $f_{KWS} = f_2 \circ f_1$ ，参数为 $\theta = (\theta_1, \theta_2) \in \Theta = (\Theta_1, \Theta_2)$ ：

$$O_K = f_{KWS}(X, K; \theta) \quad (11.4)$$

11.2 关键词唤醒和LVCSR

在前面的章节已经介绍了LVCSR（大词汇量语音识别任务）的定义和解决方法，我们来分析KWS任务和LVCSR任务的相关性。LVCSR任务是将一段连续语音识别成文字。该任务的HMM-GMM系统的原理是为每个音素进行建模，将输入语音的每帧识别成音素的状态。LVCSR任务的DNN方法输入是每帧语音信号特征向量，输出是该帧对应的音素，本质上是一个多分类问题。KWS任务可以被视为LVCSR任务的子问题，不同的是KWS任务只需要在一段连续语音中提取关键字的信息。而实际上关键词唤醒问题的解决方法和LVCSR在一定程度上是相通的，这将在后文进行详细阐明。

11.3 关键词唤醒的难点

在主观感受中，关键词唤醒任务需要识别的只是几个关键词而已，为什么会划分为一个单独的问题进行研究。因为关键词唤醒问题在实际产品落地中有很大的挑战，其中最主要的问题是低功耗和高计算需求的不平衡。现实应用关键词唤醒任务的产品如智能手机、智能音箱等都是计算量不大的低端芯片，且大都是电池供电。这就需要系统尽可能减小运算量和降低功耗，但同时不能降低系统的稳定性。这就是关键词唤醒系统设计中的难点所在。

11.4 模型方法

11.4.1 Query-by-Example方法

Query-by-Example方法[47]是解决关键词唤醒问题最早尝试的方法之一。顾名思义，query-by-example方法是一种将关键词当作example，将听到的语音（query）和关键词（example）进行比对的方法，这是一种基于模板匹配的方法。具体实现包括两个步骤：

(1)训练步骤

先选择合适的方法对将语音格式的关键词表示为易于比较的模板格式（例如特征向量或网格形式等），完成对关键词的建模。

(2)测试步骤

将待检测的目标语音转化成和模板相同的格式，然后和关键词模板进行匹配计算相似度。

在过去几十年对query-by-example方法的探索中，研究热点是如何选择合适的特征表示方法[48]对关键词进行建模，而模板比对主要采用动态时间规整（dynamic time warping）[49]的算法，它采用了动态规划的思想。在自动语音识别的孤立词识别任务中，也采用了类似query-by-example的方法。将给定的词汇创建模板，将待识别语音和模板比对，相似度最大的即为识别结果。DTW方法在孤立词识别任务中也能取得较好的效果。query-by-example方法不再是KWS问题的主流方法，因此此处不再做过多展开。

11.4.2 Keyword/Filler隐马尔科夫系统

基于隐马尔可夫模型的keyword/filler系统[50]是目前的主流方法之一，它和语音识别中早期使用的HMM-GMM系统原理相似。其思想都是用HMM模型对待识别的语音的子字单元进行建模。最大的不同是在解码器中，LVCSR方法的解码器中包含了词典中所有的单词，而关键词唤醒任务的keyword/filler系统则将语音分为关键词（keyword）和非关键词（filler）两类进行建模。如图11.1所示，keyword建模采用精细建模方法，在词级、音素级

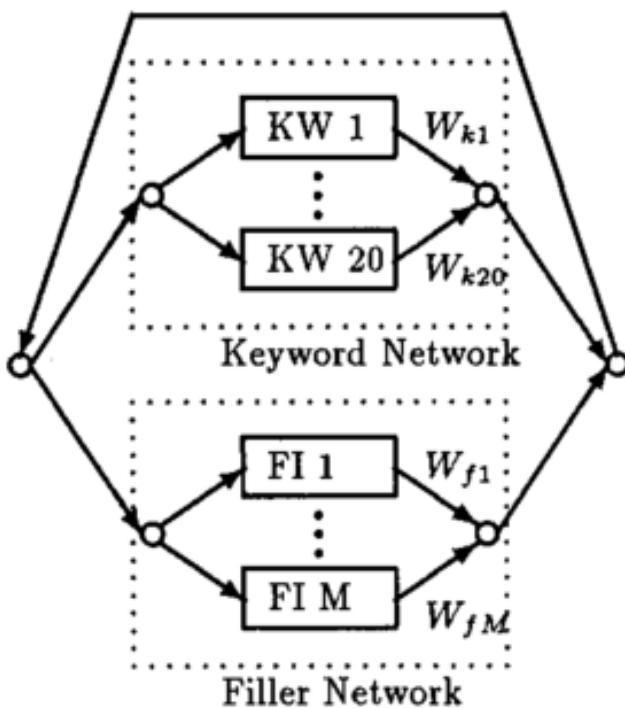


图 11.1: keyword-filler系统

或状态级上对关键词进行建模；filler建模采用粗放建模的方法，对除关键词之外的任意词语和噪音进行建模。这样建模的好处是大大缩小了解码空间，使解码速度变快，提高系统的实时率。

如图11.2所示为基于HMM系统的keyword-filler的一个实例，来自于Amazon的文章[51]。该解码系统由keyword(alex)路径和filler路径构成，filler路径又分为非语音（Non-speech,NSP）路径和语音（speech,SP）路径。对关键词采用三状态HMM对音素进行建模，而filler采用单状态HMM进行建模，有趣的是该隐马尔可夫模型的发射概率由TDNN进行建模。

11.4.3 LVCSR方法

一种简单粗暴的想法是直接用LVCSR语音识别系统解决关键词检索唤醒问题[52]。待识别语音通过LVCSR系统给识别为一个个单词或字，再进一步进行关键词检索。基

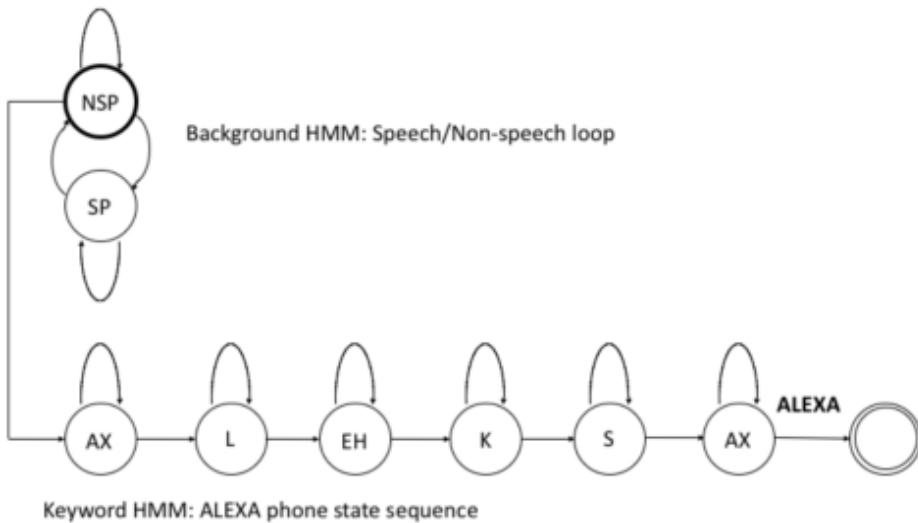


图 11.2: Alexa keyword-filler 系统

于LVCSR系统方法的一个缺点是OOV问题。如果关键词不在词典中，那KWS问题将无法进行。基于LVCSR的关键词唤醒虽然能取得较好的效果，但是计算需求量大，在训练资源充足的时候系统稳定性较强，但是在移动设备上连续运行LVCSR系统通常是不切实际的。

11.4.4 Deep KWS端到端系统

目前最为流行的方法是基于神经网络的端到端关键词唤醒系统，在论文[53]中被首次提出。如图11.3所示为deep KWS系统的具体结构。该系统分为三个模块：

- (1) 特征提取模块。对原始语音信号进行特征提取，通常做法是提取每帧的特征向量。
- (2) 神经网络模块。输入是语音的特征向量。输出是关键词和非关键词的后验概率。
- (3) 后验值处理模块。由于神经网络输出的后验值是带有噪音的。在这一步骤中，对后验值以一定窗长进行平滑。平滑后的后验值超过一定的阈值，则认为被唤醒。

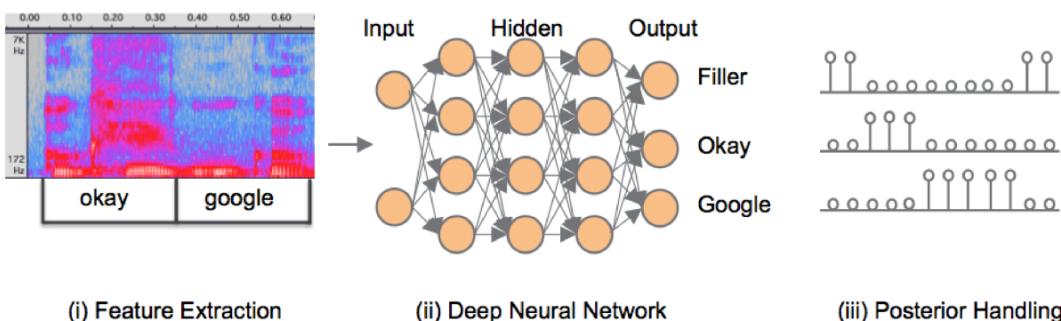


图 11.3: Deep KWS系统框架，从左到右依次为特征提取，神经网络分类器和后验值处理。

11.5 关键词唤醒和嵌入式系统

我们知道关键词唤醒系统的应用场景大都是计算能力不大的设备。比如可穿戴设备，车载语音系统，家居语音助手等，在这种应用场景中，如何用深度学习的方法解决实际问题仍待完善。

目前在嵌入式设备上的关键词唤醒系统优化主要分为两类：一是通过减少神经网络参数数量的方法提高系统在设备上的性能。二是针对硬件平台的优化。

11.5.1 减少模型参数量

模型裁剪

当训练好神经网络模型时，我们会发现有些神经元连接的权重值很小，在这种情况下可以认为两个神经元的关联不是很大，进行裁剪处理，从而得到较小的神经网络，减小网络模型的复杂度。

模型压缩

针对神经网络压缩常见的方法是SVD方法。通过奇异值分解将权重矩阵分解成两个较小的矩阵相乘，从而减小模型的计算量，提高系统在设备上的性能。

11.5.2 针对硬件平台的优化

目前嵌入式端设备的主流处理器有X86，ARM，MIPS等。当神经网络在服务器上训练好后，部署到嵌入式平台可以有针对性地使用运算加速库提高系统性能。常用的运算加速库有ATLAS,OpenBLAS,MKL等。

11.5.3 其他优化方法

在嵌入式系统工程应用当中，还采用了很多其他的优化方法。比如采用跳帧（subsampling）操作可以使速度提升2-3倍，同时准确率损失很小；在关键词唤醒系统中采用VAD过滤的方法，过滤非语音信号的音频，减少运算量的同时，还能降低系统的虚警率，提高系统准确性和灵敏度。

11.6 小结

本节介绍了关键词唤醒的概念、技术难点和解决方法。模型方法包括Query-by-Example通过DTW算法进行模板匹配的方法；Keyword/Filler系统基于隐马尔可夫模型对关键词和非关键词进行建模的方法；基于LVCSR的关键词唤醒方法以及Deep KWS端到端输出后验概率的方法。目前工程应用上多采用Keyword/Filler系统和Deep KWS系统，在保证系统准确率的同时降低系统能耗并提升系统的反应速度。

关于关键词唤醒系统在嵌入式设备端的部署，主要介绍了两种方法。一是通过减少模型本身的复杂度降低运算量，二是通过运算加速库针对硬件平台进行优化，提高运算速度。

前沿课题



12	说话人识别	85
13	语种识别	87
14	情绪识别	89
15	语音合成	91
15.1	激励-响应模型	
15.2	参数合成	
15.3	拼接合成	
15.4	统计模型合成	
15.5	神经模型合成	
15.6	基于注意力机制的合成系统	
15.7	小结	

12. 说话人识别

13. 语种识别

14. 情绪识别

15. 语音合成

by 启明¹

语音合成是由文字生成声音的过程，通俗地说，就是让机器按人的指令发出声音。早在1769年，匈牙利发明家Wolfgang von Kempelen就建造了一台会说话的机器。这台机器用机械装置模拟人的发音机理，通过风箱驱动簧片产生声音。1845年奥地利发明家Joseph Faber发明了Euphonnia，通过键盘可以发出声音。这些早期发声机器用机械装置模拟人的发音过程，清晰度较低，只能发一些简单音素和单词。

计算机发明以后，语音合成技术开始快速发展。按时间顺序，语音合成方法可归纳为四种：参数合成、拼接合成、统计模型合成和神经模型合成。绝大多数合成方法都基于激励-响应（Source-Filter）模型，我们首先对该模型做简单介绍。

15.1 激励-响应模型

人类的发音可以描述为一个声门激励序列经过一个声道响应函数的滤波过程，称为激励-响应模型。基于这一模型，声音是由肺部气流冲击声门（主要是声带）产生振动，再通过口鼻所形成的声音共鸣产生。如果声带完全打开，则振动源是一个随机噪声，这时产生的声音是清辅音；如果声带拉紧，声门生成周期性振动，这时产生的声音是浊辅音和元音。在这一模型中，气流冲击声带称为声门激励，声道的共鸣称为声道响应，是对声门激励信号的滤波或调制。图15.2给出了这一语音生成过程。

我们以元音为例观察上述声道-激励模型产生的语音信号的特点。基于该模型，语音信号是声门发出的基础周期信号经过声道的共鸣。这一过程有两个效果：一方面，原始周期信号在声道所形成的共振腔中反射叠加，会生成一系列倍频信号，在频域上形成

¹wangd.csit.org



图 15.1: Saarland University 大学于 2007-2009 年重现的 Kempelen 发声器。

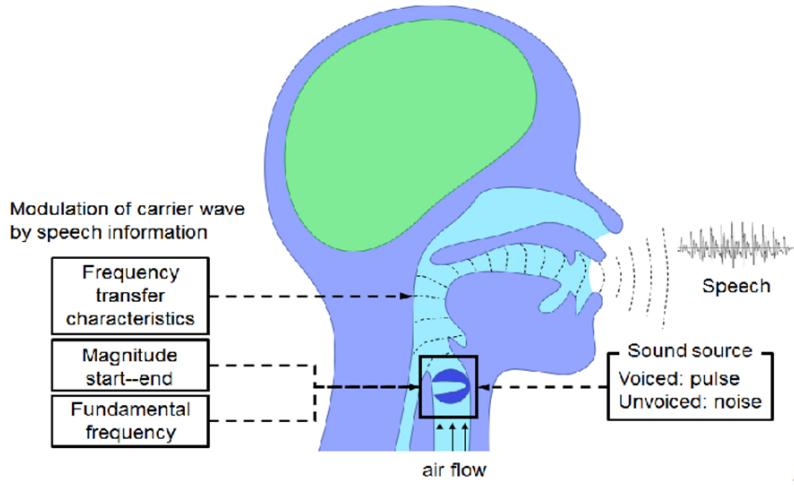


图 15.2: 人类发音的声门激励-声道响应模型。气流冲击声门产生混乱的或周期性的振动，这一振动通过口鼻形成的声音形成共鸣和调制。

周期性波动；另一方面，声道具有调制作用，对不同倍频信号产生不同的增益，形成频谱上的包络。图 15.3 给出语音信号的频谱特性，其中 F_0 是原始周期信号的频率，称为基频， F_1, F_2, F_3 分别为频谱包络的极大值，称为第一、第二、第三共振峰。需要强调的是， F_0 表征的是声门的激励特性，而各个共振峰反应的声道传递函数，二者具有完全不同的物理意义。

激励-响应模型是一个生成模型，给定基频和各个共振峰，即可计算出声门激励和声道的传递函数，将二者通过卷积即可以合成功音，另一方面，基于该模型，也可以实现语音信号的分解，将语音分解为声门信号和声道响应。这一分解事实上是一个解卷积过程。最著名的分解方法是基于线性预测（LPC）的分解，即假设语音信号具有线性可预测性，

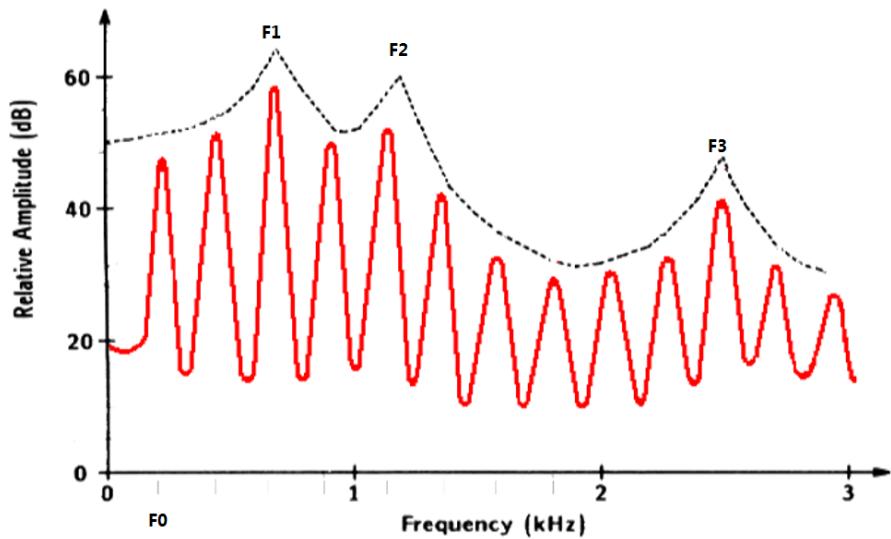


图 15.3: 语音信号是倍频后的声门激励和声道特性的卷积，在频域上表现为声道特性对不同频率幅值的调制。图中红色实线表示实际频谱，黑色虚线为频谱包络，代表声道的响应函数。

预测系数构成了声道特性，预测的残差即对应声门激励。事实上，这一假设对应一个声道的全极点模型，LPC事实上是对该模型极点位置的估计。图15.4给出这一分解的示意图。

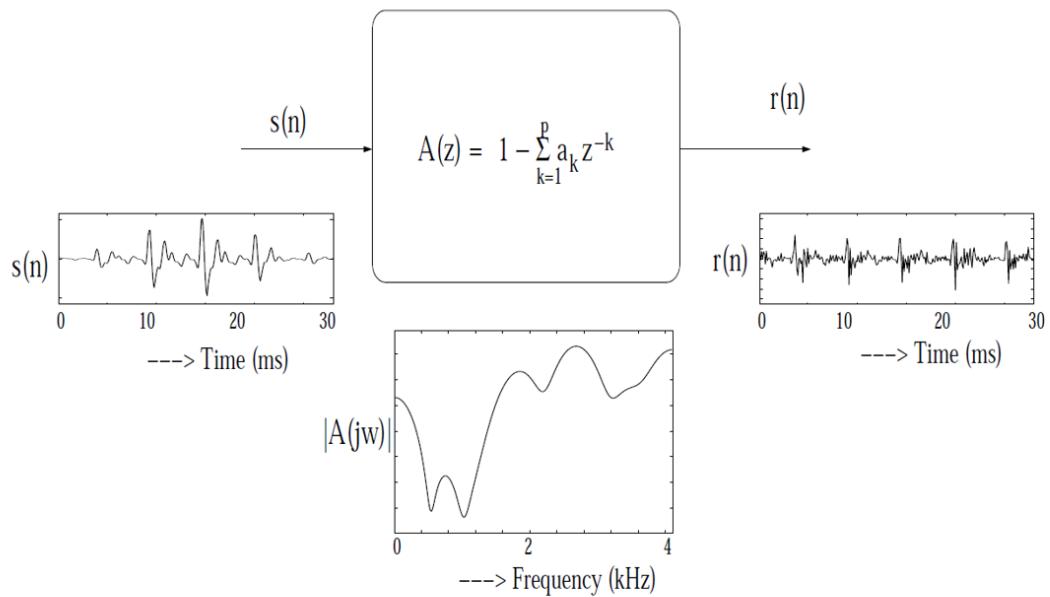


图 15.4: 基于线性回归模型 (LPC) 的语音信号分解。

我们已经看到，基于激励-响应模型，我们既可以对语音进行分解，也可以基于分解得到的成分实现对声音的合成。对声音进行分解-合成的装置称为声码器，其中分解部分

称为编码器，合成部分称为解码器。历史上第一个声码器由贝尔实验室于1930年发明，这一发明奠定了语音合成研究的基础。图15.5给出了一个声码器中的编-解码流程示意图，其中编码器将声音分解成基频和频谱包络两部分，前者代表声门特性，后者代表声道特性。这两部分信号经过通信信道传送到解码器，解码器利用这两路信息，基于激励-响应模型合成语音。

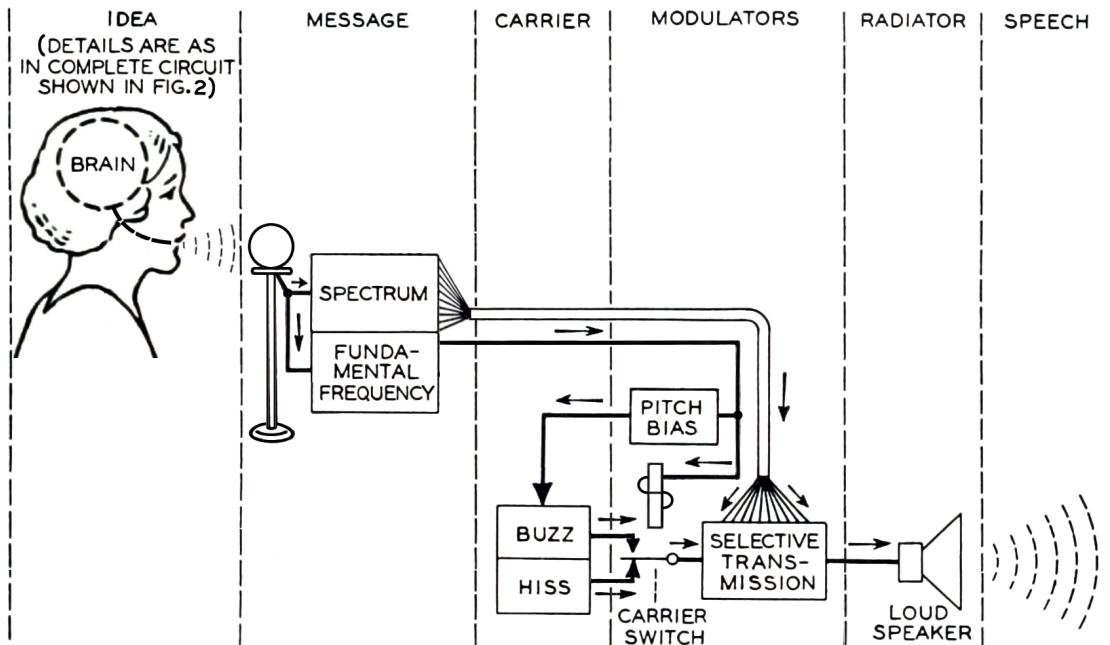


图 15.5: 声码器的基本流程。

15.2 参数合成

由前一节所介绍的声道-激励模型可知，语音信号可分解为声门激励和声道响应两部分，二者共同决定了发出声音的特性。因此，只要设计合适的激励信号和合适的声道响应函数，即可合成目标声音。

前面我们已经提到，声门激励与发音的类型有关，如辅音和元音，而声道特性直接决定发音的内容。图15.6给出不同元音在共振峰平面（F1-F2）上的分布情况，可以看到不同元音在这一平面上有显著区分，意味着我们只要指定F1和F2的值，即可生成对应的元音。在实际系统中，可以对现有发音库进行分析，统计不同音素（包括其上下文）的基础和各个共振峰的取值，在合成时利用这些取值即可合成出相应的声音。这一方法称为参数合成。

DEC公司推出的DECTalk是这种合成方法的典型代表。参数合成的优点是计算量小，可调节性高，缺点是参数调节困难，生成的声音自然度较低。

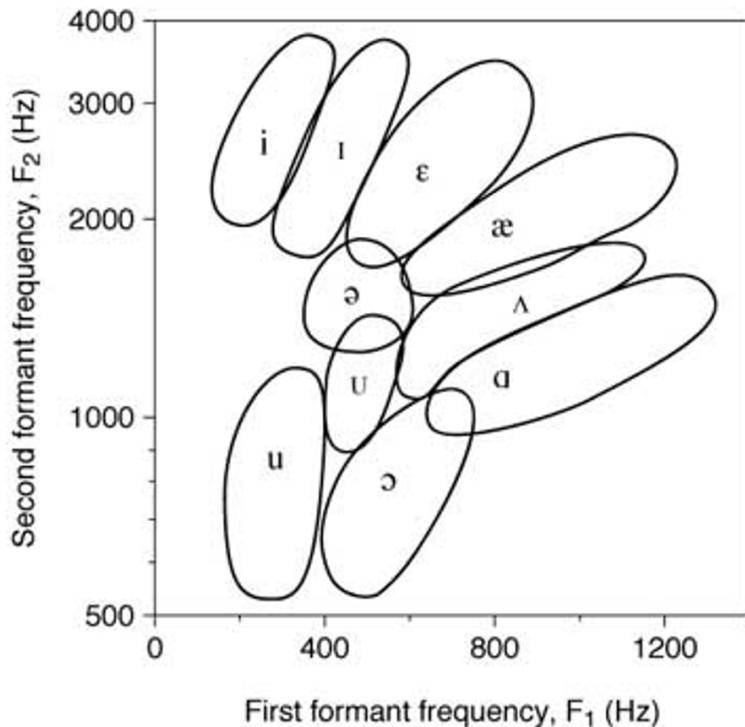


图 15.6: 不同元音在F1-F2平面上的分布。

DECTalk lets micros read messages over phones

BY PEGGY ZIENTARA
Senior Editor

The use of the telephone as a "universal terminal" came one step closer to reality when Digital Equipment Corporation (DEC) introduced a voice synthesizer unit that can "read" aloud ASCII text **messages**.

DECTalk, which the company says will be available in March at a price of \$4000, allows you to call in to a personal computer from anywhere in the world by touch-tone telephone (including pay phones), to hear electronic mail **messages** stored in memory.

DECTalk can also "tell" you what you just typed into memory on the computer keyboard if you simply press a punctuation mark key and the Return key.

Business applications are expected to include access to company data bases by traveling sales representatives; access by lawyers to public and private computer files during trial breaks; direct customer transactions with banks and stock brokerage firms; direct catalog and retail sales; and access to transportation schedules, according to Ed Kramer, vice-president of corporate marketing for DEC.

Other applications will include aid for the speech-impaired (now in place at

via any touch-tone phone.

The DECTalk hardware, which fits easily under a 12-inch monitor, will feature eight different types of voices, including that of an old man, a deep-voiced man, a typical middle-aged man, two different women and a child. The unit attaches to a personal computer via an RS-232C serial interface.

DEC claims to have licked the traditional voice technology trade-offs of hav-

ing either good voices with limited vocabularies or unlimited vocabularies with poor voices. DEC's unit has a "virtually unlimited vocabulary," Kramer said, and the speaking rate can be adjusted from 120 to 350 words per minute. Tone and modulation can also be regulated.

Heuristics — interpreting word context to improve pronunciation — and a computer model of the human vocal tract allow DECTalk to produce high-quality voices, in contrast with traditional stored digitized or synthesized voice technology, Kramer explained.

The unit accepts ASCII text, which it applies to an internal dictionary and a set of letter-to-sound rules indicating pronunciation and context interpretation to



图 15.7: 1984年InfoWorld报导DECTalk。DECTalk一直是著名科学家霍金的发音助手。

15.3 拼接合成

上个世纪九十年代，随着大规模语音库的积累，基于拼接的合成方法成为主流。这一方法将事先录好的语音切分成发音片段（一般为音素），在合成时从这些片段中选择合适的候选进行拼接组成句子 [54]。

拼接法需要处理两个主要问题。第一个问题是对于上下文环境的处理，同一个音素在不同环境下发音会有所差别，因此需要根据环境选择合理的发音片段。这里的环境既包括前后音素，也可以包括词边界、词性、句子属性等。第二个问题是拼接连贯性问题。把两个发音片段拼在一起时，总会产生一些不连续性。一般采用在频域进行拼接和平滑的方法来增加连贯性。

早期的拼接合成多采用半音素（di-phone）作为发音片段。所谓Di-Phone，是两个相邻音素对中从一个音素的中心到另一个音素的中心对应的发音片段，如图15.8所示。半音素模型有若干好处，一是在进行语音标注时，标注音素的中心要比标注音素边界容易的多；二是半音素中包含了音素音的变换过程，有利于描述一定的上下文关系；三是音素中心一般比较稳定，在这里拼接有利于保持发音的连续性。

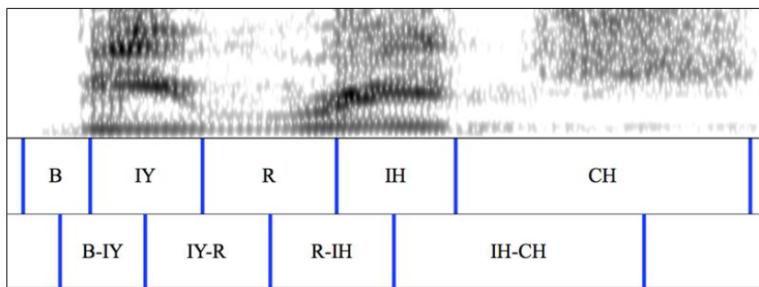


图 15.8: 语音流中的半音素 (Di-Phone)，其中B-IY, IY-R等都是半音素。

另一种拼接合成方法是基于单元选择 (Unit Selection) 的方法，如图 15.9 所示。在这种方法里，每个发音单元是一个上下文相关音素，不同的音素可能有多个单元，每个单元拥有包括环境相关的各种标记，以及相应的基频、频谱包络等。在合成时，将目标句子的上下文环境选择合适的单元，将相应的基频和频谱包络拼接起来。在单元选择过程中，需要考虑两个准则，一是使得选择出的单元符合环境约束，二是使得选择出的单元在互相拼接时保持连贯性。和半音素方法相比，单元选择方法基于更大规模的数据库，对环境的建模更细致。理论上说，如果数据库规模足够大，我们总可以选出合适的单元，生成自然流畅的声音。

15.4 统计模型合成

基于拼接的合成方法可以生成高质量的语音，但需要较大规模的语音库，录制成本较高，占用磁盘空间较大，在嵌入式设备上很难实用。另外，拼接方法无法灵活地改变声音的特性，在应用上有一定局限性。

为解决拼接方法的这一困难，人们提出统计模型方法。该方法为每个发音单元建立一个统计模型，在合成时仅利用这些模型进行生成。由于在合成时不需要语音库，基于统计模型的合成系统通常比较精简 [55]。特别重要的是，基于统计模型可以很容易实现对合成语音的控制，这对合成个性化的声音尤为重要。基于隐马尔可夫模型 (HMM) 的统计模型方法是这一时期的主流。这种方法对每个发音单元建立一个 HMM 模型，在合成时将句子中所有音素的 HMM 模型拼接起来形成一个组合模型，由该模型生成最匹配的语音。

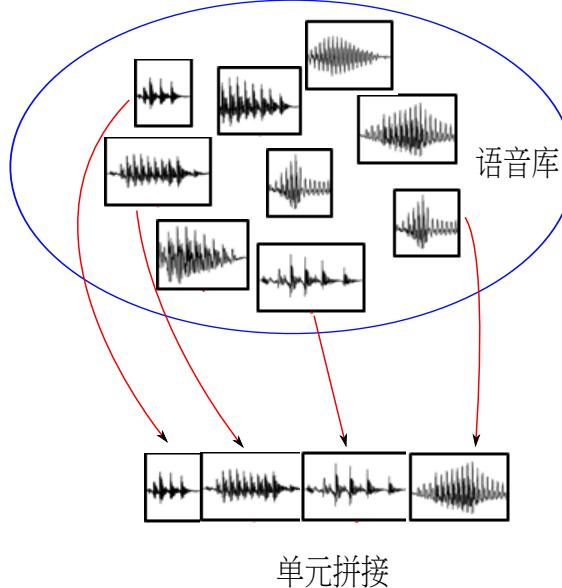


图 15.9: 基于单元选择的语音合成。合成时从语音库中选择合适的发音片段进行拼接。选择发音片段时需要考虑该片段是否符合上下文约束的发音要求，还需要考虑是否可以和其它片段实现自然拼接。图片设计基于HTS slides。³

具体而言，我们首先对语音信号进行切分，生成上下文相关音素，再对每个音素的长度、基频和频谱包络建立三个HMM模型。上下文环境信息一般包括前后各两个音素标识、词内位置、句内位置、词性、韵律信息等。长度模型是一个单一状态单个高斯的HMM，基频和频谱包络是多状态HMM。基频的状态概率分为两个空间，一个离散空间用来描述清辅音的零基频，一个连续空间用来描述浊辅音和元音的非零基频。频谱包络的HMM的状态概率模型是一个高斯混合模型（GMM），用来描述频谱包络在该状态的分布规律。

在实际合成时，首先需要基于长度模型选择每个音素的发音长度，之后基于基频模型和频谱包络模型生成每个音素的基频和频谱包络，最后基于生成的基频和频谱包络合成语音。在生成基频和频谱包络时，生成准则是使得生成基频和频谱包络在对应HMM中的输出概率最大化。如果不考虑语音帧之间的相关性时，则每个音素中每个状态在不同时刻的输出是不变的。显然，这一输出会在状态的分界面上产生跳跃，影响合成的质量。为解决这一问题，研究者提出基于动态特征建模的方法。基于这一模型，每个状态的输出需要考虑前后帧的相关性，这一约束使得生成的基频和频谱包络更加连续。

图 15.10给出对频谱包络中的某一维进行生成的过程，其中红色折线不考虑前后帧相关性时的最大概率输出；当考虑前后帧约束时，即生成更为连续的输出，如图中蓝色曲线所示。

图15.11给出HMM合成系统的总体框架。在训练阶段，对数据库中的语音信号进行分析，分解成基频（对应声门）和频谱包络（对应声道响应），利用数据库中的标注信息，对每个上下文相关音素建立时长、基频和频谱包络的HMM模型。在合成阶段，首先对输入文本进行分析（一般称为前端处理），将文本转换成标注文件，基于该标注选择相应的音素，并将这些音素对应的基频HMM和频谱包络串连起来，生成句子的基频和频谱包络

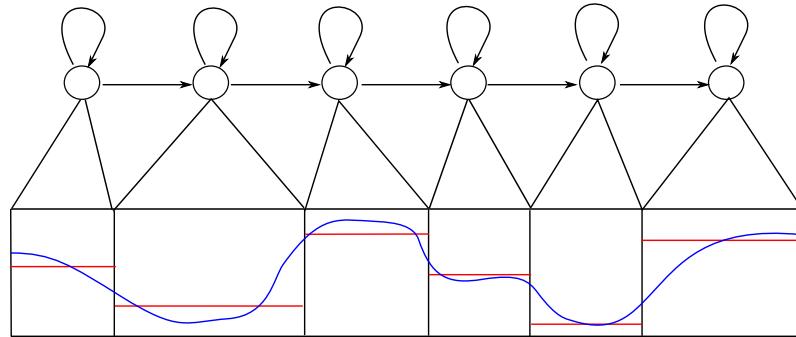


图 15.10: 基于HMM模型生成某一维频谱包络的过程。每个蓝色圆圈代表一个HMM状态，虚线箭头标出每个状态要生成的语音片段。下方两条水平平行线间的竖线表示每一帧的生成值。实际生成时，首先估计出每个音素中每个状态要生成的帧数，再对每个状态进行生成（图中红色折线）。考虑到前后帧的相关性，应对生成的语音帧进行平滑，即得到图中的蓝色生成曲线。图片设计基于HTS slides。

序列，最后送入声码器得到合成语音。

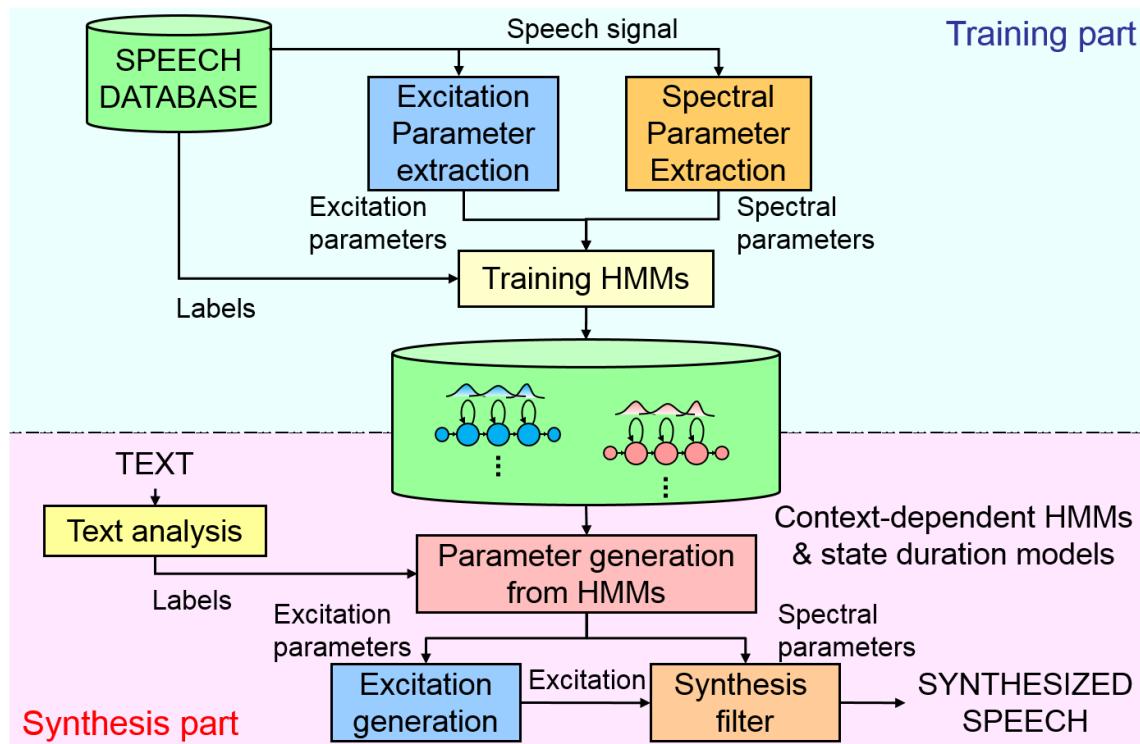


图 15.11: 基于HMM的语音合成系统框架图。图片来自HTS slides。

15.5 神经模型合成

在讨论语音识别时我们已经知到，HMM模型不能描述复杂的发音现象。当数据量增加时，基于HMM的语音合成系统的性能遇到了瓶颈。为此，研究者提出基于深度神经网

络（DNN）的语音合成方法。这一工作由香港中文大学、微软、Google于2013年提出 [56, 57, 58, 59]。在这些模型中，研究者用神经网络取代HMM模型来预测每一帧语音的激励和调制信号，再通过声码器合成自然语音。

图15.12是Google的DNN合成系统示意图。该系统的前端处理部分和HMM系统是一样的，不同的是用DNN取代HMM来生成每个音素的时长、基频和频谱包络。事实上，Google系统中生成的是这些量的均值和方差，基于这些统计量，可利用HMM系统类似的方法，通过考虑前后帧之间的相关性生成每一帧的实际参数，即图中的“Parameter Generation”部分。生成的参数最后还是要通过一个声码器来合成语音。

2014年以后，研究者对基于DNN的合成方法进行了一系列扩展，提出了基于RNN的合成系统。和DNN相比，RNN可以学习长时相关性，因此可以用来学习发音过程中前后音素之间的协同发音特性，从而得到更平滑自然的发音。图15.13是微软发表的基于RNN的合成系统示意图 [60]。该模型首先生成每个音素的长度，然后基于一个双向RNN模型对每个语音帧直接预测基频和频谱包络参数。因此，该系统并不需要一个额外的参数生成器，由音素参数的统计量生成每帧的参数。

15.6 基于注意力机制的合成系统

近两年来，基于神经模型的语音合成取得长足进展，其中基于注意力机制的语音合成系统最受瞩目 [61]。直观上，基于注意力机制的语音合成可类比人类的阅读-理解-复述过程。首先，用一个双向RNN将要发音的文本编码，这类似于阅读和理解；之后基于另一个RNN逐一生成每个语音帧，类似于对脑海中的记忆进行复述。在每一步生成时，系统基于注意力机制定位到要发音的文本，并利用该文本的信息指导生成过程。

图15.14是Google基于该思路设计的Tacotron系统。在该系统中，待合成的句子以字符串的形式输入到一个编码器，生成一个完整的句子编码。在合成时，基于一个RNN迭代合成每一帧参数，每合成一帧的时候，不仅将前一帧输出作为输入，同时基于Attention机制提取句子编码中相应的信息，使得生成的语音与输入要求相一致。特别值得说明的是，这一方法的解码输出并不是基频和频谱包络，而是直接生成频谱。基于频谱可以直接合成原始语音，而不再需要一个声码器。一种常用的由频谱合成语音的算法是Griffin-Lim算法，该方法首先由频谱估计出相位，再将相信应用到频谱上，合成语音。Griffin-Lim算法简单方便，但性能不高。Google提出利用多层卷积网络由频谱直接合成语音，这一网络称为WaveNet[62]，如图15.15所示。

Tacotron已经成为语音合成系统的主流模型，可以合成非常自然的声音。人们对这一模型进行了一系列扩展，例如，DeepMind提出了一种并行WaveNet算法，可以使WaveNet的合成速度达到实时 [63]，Google将说话人信息表达成一个说话人向量，作为辅助输入，使得Tacotron系统可以合成不同人的声音[64]。

15.7 小结

本节我们简要介绍了语音合成的基本方法。绝大部分合成方法基于激励-调制模型，

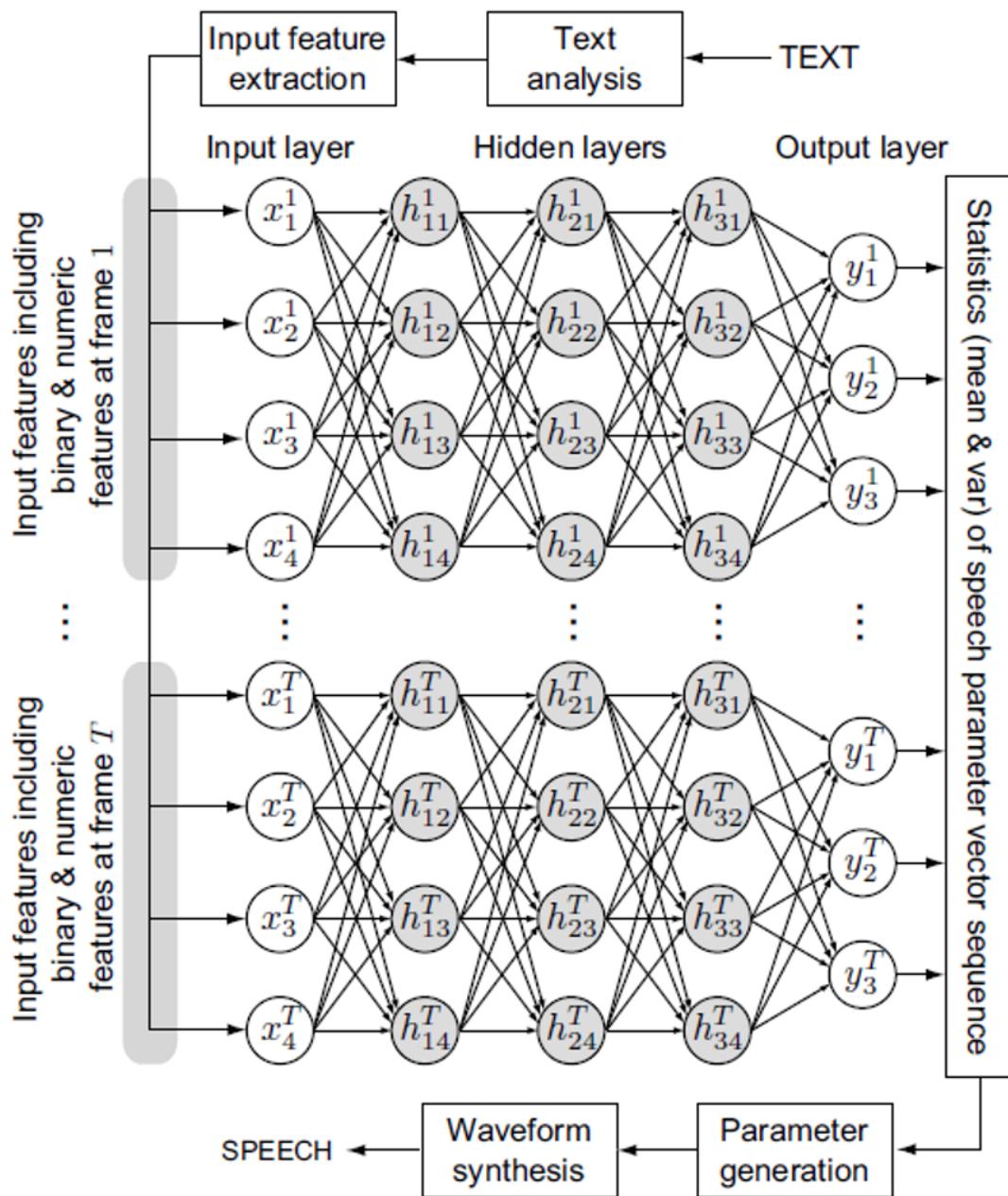


图 15.12: Google 基于DNN的语音合成系统。该模型通过DNN来预测每个音素的时长、基频和频谱包络的均值和方差，基于此生成每帧的参数，最后送入声码器合成语音。图片设计源自 [59]。

该模型认为语音是由声门激励经过声道的调制过程，因此，只要设计合适的激励和声道响应函数，即可合成需要的声音。最早的语音合成系统基于共振峰参数生成声道响应函数，基于此合成功能。这一方法效率很高，但由于参数过于简单，质量不能保证。拼接法从大规模数据库中直接提取声门和声道参数，可以实现较逼真的合成，但在拼接点有不连续情况，且占用系统资源较多。统计模型法是一种对共振峰合成的回归，但利用了更

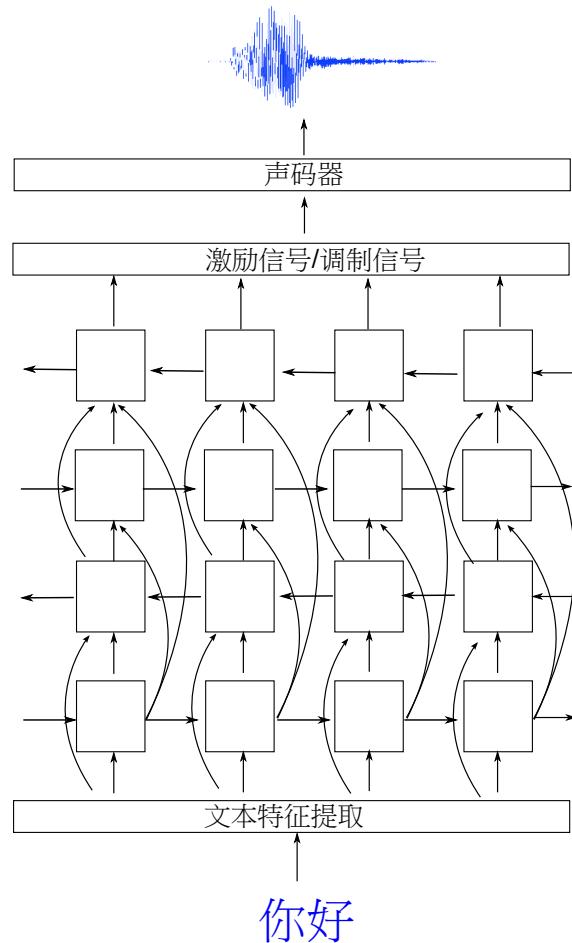


图 15.13: 微软基于RNN的语音合成过程。该模型基于多层双向RNN对输入句子进行映射，预测每一帧语音的激励和声道调制信号，最后通过声码器合成声音。图片设计源自 [60]。

多信息和更复杂的概率模型对声门和声道的参数进行预测。神经模型法用更复杂的神经网络代替HMM等统计模型，当训练数据量较大时可以突破统计模型在概率形式上的限制，因而合成更自然流畅的声音。近两年来，基于神经模型的合成系统进一步发展，特别是以Tacotron为代表的端对端系统，将待合成的句子直接映射到频谱，事实上已经摆脱了传统合成方法中对激励-调制模型的依赖，成为新一代合成系统。

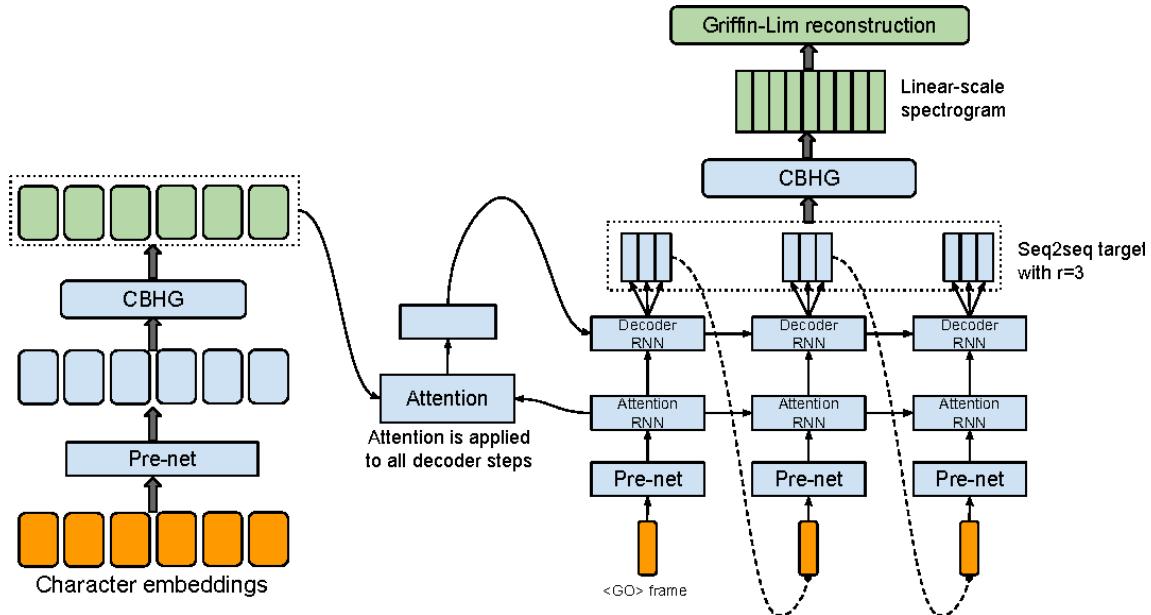


图 15.14: 基于注意力机制的Tacotron合成系统。图片源自 [61]。

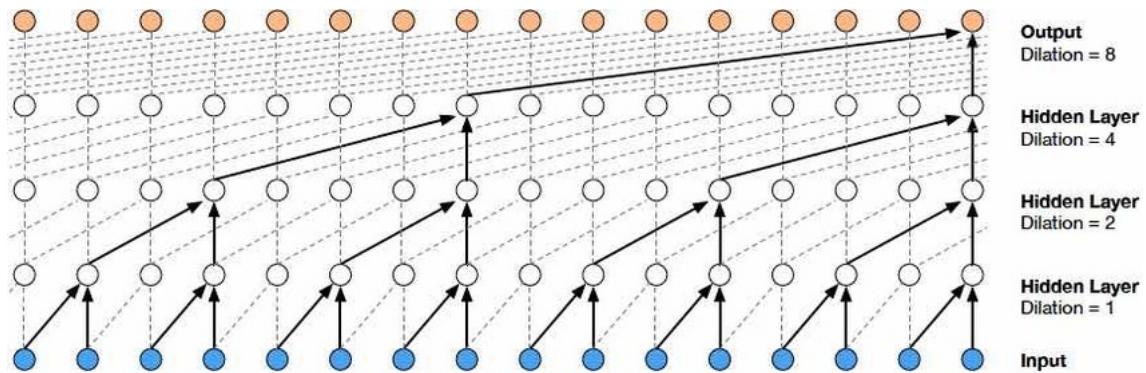


图 15.15: WaveNet结构。图片源自 [62]。



Roman Forum

16	技术收藏夹	105
17	Q & A	107
	参考文献	109
	索引	115

16. 技术收藏夹

17. Q & A

参考文献

- [1] Mehryar Mohri, Fernando Pereira, and Michael Riley. “Weighted finite-state transducers in speech recognition”. In: *Computer Speech & Language* 16.1 (2002), pages 69–88 (cited on pages 16, 40).
- [2] Jan K Chorowski et al. “Attention-based models for speech recognition”. In: *Advances in Neural Information Processing Systems*. 2015, pages 577–585 (cited on page 17).
- [3] Dzmitry Bahdanau et al. “End-to-end attention-based large vocabulary speech recognition”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pages 4945–4949 (cited on page 17).
- [4] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech recognition with deep recurrent neural networks”. In: *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE. 2013, pages 6645–6649 (cited on page 17).
- [5] Daniel Povey et al. “The Kaldi speech recognition toolkit”. In: *IEEE 2011 workshop on automatic speech recognition and understanding*. EPFL-CONF-192584. IEEE Signal Processing Society. 2011 (cited on page 19).
- [6] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. “A time delay neural network architecture for efficient modeling of long temporal contexts”. In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015 (cited on page 42).
- [7] Dong Wang and Thomas Fang Zheng. “Transfer learning for speech and language processing”. In: *arXiv preprint arXiv:1511.06066* (2015) (cited on page 48).

- [8] Li Lee and Richard Rose. “A frequency warping approach to speaker normalization”. In: *IEEE Transactions on speech and audio processing* 6.1 (1998), pages 49–60 (cited on page 48).
- [9] D Rama Sanand, D Dinesh Kumar, and Srinivasan Umesh. “Linear transformation approach to vtln using dynamic frequency warping”. In: *Eighth Annual Conference of the International Speech Communication Association*. 2007 (cited on page 48).
- [10] Xiaodong Cui and Abeer Alwan. “MLLR-like speaker adaptation based on linearization of VTLN with MFCC features”. In: *Ninth European Conference on Speech Communication and Technology*. 2005 (cited on page 48).
- [11] Jean-Luc Gauvain and Chin-Hui Lee. “Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains”. In: *IEEE transactions on speech and audio processing* 2.2 (1994), pages 291–298 (cited on page 50).
- [12] Christopher J Leggetter and Philip C Woodland. “Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models”. In: *Computer speech & language* 9.2 (1995), pages 171–185 (cited on page 50).
- [13] Mark JF Gales et al. “Maximum likelihood linear transformations for HMM-based speech recognition”. In: *Computer speech & language* 12.2 (1998), pages 75–98 (cited on page 50).
- [14] Tasos Anastasakos, John McDonough, and John 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*. Volume 2. IEEE. 1997, pages 1043–1046 (cited on page 53).
- [15] Hank Liao. “Speaker adaptation of context dependent deep neural networks”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pages 7947–7951 (cited on page 54).
- [16] Jian Xue et al. “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, pages 6359–6363 (cited on page 54).
- [17] Shaofei Xue et al. “Speaker adaptation of hybrid NN/HMM model for speech recognition based on singular value decomposition”. In: *Journal of Signal Processing Systems* 82.2 (2016), pages 175–185 (cited on page 54).
- [18] Dong Yu et al. “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, pages 7893–7897 (cited on page 55).
- [19] George Saon et al. “Speaker adaptation of neural network acoustic models using i-vectors.” In: *ASRU*. 2013, pages 55–59 (cited on page 55).

-
- [20] Andreas Stolcke. “SRILM-an extensible language modeling toolkit”. In: *Seventh international conference on spoken language processing*. 2002 (cited on page 56).
 - [21] Min Ma et al. “Approaches for neural-network language model adaptation”. In: *Proc. Inter-speech, Stockholm, Sweden* (2017), pages 259–263 (cited on page 56).
 - [22] Steven Boll. “Suppression of acoustic noise in speech using spectral subtraction”. In: *IEEE Transactions on acoustics, speech, and signal processing* 27.2 (1979), pages 113–120 (cited on page 58).
 - [23] Michael Berouti, Richard Schwartz, and John Makhoul. “Enhancement of speech corrupted by acoustic noise”. In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP’79*. Volume 4. IEEE. 1979, pages 208–211 (cited on page 59).
 - [24] Bradford W Gillespie and Les E Atlas. “Acoustic diversity for improved speech recognition in reverberant environments”. In: *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*. Volume 1. IEEE. 2002, pages I–557 (cited on page 59).
 - [25] Masato Miyoshi and Yutaka Kaneda. “Inverse filtering of room acoustics”. In: *IEEE Transactions on acoustics, speech, and signal processing* 36.2 (1988), pages 145–152 (cited on page 59).
 - [26] Bayya Yegnanarayana and P Satyanarayana Murthy. “Enhancement of reverberant speech using LP residual signal”. In: *IEEE Transactions on Speech and Audio Processing* 8.3 (2000), pages 267–281 (cited on page 59).
 - [27] Tomohiro Nakatani et al. “Blind speech dereverberation with multi-channel linear prediction based on short time Fourier transform representation”. In: *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE. 2008, pages 85–88 (cited on page 59).
 - [28] Jacob Benesty, Jingdong Chen, and Yiteng Huang. *Microphone array signal processing*. Springer Science & Business Media, 2008 (cited on page 59).
 - [29] Kenichi Kumatani, John McDonough, and Bhiksha Raj. “Microphone array processing for distant speech recognition: From close-talking microphones to far-field sensors”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pages 127–140 (cited on page 59).
 - [30] Michael L Seltzer, Bhiksha Raj, Richard M Stern, et al. “Likelihood-maximizing beam-forming for robust hands-free speech recognition”. In: *IEEE Transactions on Audio, Speech and Language* 12.5 (2004) (cited on page 61).
 - [31] Bishnu S Atal. “Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification”. In: *the Journal of the Acoustical Society of America* 55.6 (1974), pages 1304–1312 (cited on page 62).

- [32] Angel De La Torre et al. “Histogram equalization of speech representation for robust speech recognition”. In: *IEEE Transactions on Speech and Audio Processing* 13.3 (2005), pages 355–366 (cited on page 63).
- [33] Hynek Hermansky and Nelson Morgan. “RASTA processing of speech”. In: *IEEE transactions on speech and audio processing* 2.4 (1994), pages 578–589 (cited on page 63).
- [34] J.Droppo and A. Acero. “Environmental Robustness”. In: *Springer handbook of speech processing*. Edited by Jacob Benesty, M Mohan Sondhi, and Yiteng Huang. Springer, 2007 (cited on page 63).
- [35] Xue Feng, Yaodong Zhang, and James Glass. “Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE. 2014, pages 1759–1763 (cited on page 64).
- [36] Kun Han et al. “Learning spectral mapping for speech dereverberation and denoising”. In: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 23.6 (2015), pages 982–992 (cited on page 64).
- [37] Bo Wu et al. “A reverberation-time-aware approach to speech dereverberation based on deep neural networks”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.1 (2017), pages 102–111 (cited on page 64).
- [38] Mengyuan Zhao et al. “Music removal by convolutional denoising autoencoder in speech recognition”. In: *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific*. IEEE. 2015, pages 338–341 (cited on page 64).
- [39] Mark John Francis Gales. “Model-based techniques for noise robust speech recognition”. PhD thesis. University of Cambridge Cambridge, 1995 (cited on page 65).
- [40] Pedro J Moreno, Bhiksha Raj, and Richard M Stern. “A vector Taylor series approach for environment-independent speech recognition”. In: *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*. Volume 2. IEEE. 1996, pages 733–736 (cited on page 65).
- [41] Zhiyuan Tang, Dong Wang, and Zhiyong Zhang. “Recurrent neural network training with dark knowledge transfer”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pages 5900–5904 (cited on page 66).
- [42] Dong Yu et al. “Feature learning in deep neural networks-studies on speech recognition tasks”. In: *arXiv preprint arXiv:1301.3605* (2013) (cited on page 67).
- [43] Shi Yin et al. “Noisy training for deep neural networks in speech recognition”. In: *EURASIP Journal on Audio, Speech, and Music Processing* 2015.1 (2015), page 2 (cited on page 67).

-
- [44] Chanwoo Kim et al. “Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home”. In: *Interspeech 2017*. 2017 (cited on page 67).
 - [45] Tom Ko et al. “A study on data augmentation of reverberant speech for robust speech recognition”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE. 2017, pages 5220–5224 (cited on page 67).
 - [46] Guoguo Chen. “Low Resource Keyword Spotting”. In: *Department of Electrical and Computer Engineering Johns Hopkins University Baltimore, Maryland* (2014) (cited on page 78).
 - [47] John S Bridle. “An efficient elastic-template method for detecting given words in running speech”. In: *Brit. Acoust. Soc. Meeting*. 1973, pages 1–4 (cited on page 79).
 - [48] Petr Fousek and Hynek Hermansky. “Towards ASR based on hierarchical posterior-based keyword recognition”. In: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. Volume 1. IEEE. 2006, pages I–I (cited on page 79).
 - [49] Hiroaki Sakoe and Seibi Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE transactions on acoustics, speech, and signal processing* 26.1 (1978), pages 43–49 (cited on page 79).
 - [50] Richard C Rose and Douglas B Paul. “A hidden Markov model based keyword recognition system”. In: *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*. IEEE. 1990, pages 129–132 (cited on page 80).
 - [51] Ming Sun et al. “Compressed time delay neural network for small-footprint keyword spotting”. In: *Proc. Interspeech*. 2017, pages 3607–3611 (cited on page 80).
 - [52] John S Garofolo, Cedric GP Auzanne, and Ellen M Voorhees. “The TREC spoken document retrieval track: A success story”. In: *Content-Based Multimedia Information Access-Volume 1*. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE. 2000, pages 1–20 (cited on page 80).
 - [53] Guoguo Chen, Carolina Parada, and Georg Heigold. “Small-footprint keyword spotting using deep neural networks.” In: *ICASSP*. Volume 14. Citeseer. 2014, pages 4087–4091 (cited on page 81).
 - [54] Alan W Black. “Perfect synthesis for all of the people all of the time”. In: *Proceedings of 2002 IEEE Workshop on Speech Synthesis*. IEEE. 2002, pages 167–170.
 - [55] K Tokuda et al. “The HMM-based speech synthesis system”. In: *Online: http://hts. ic.nitech. ac. jp* (2007).
 - [56] Shiyin Kang, Xiaojun Qian, and Helen Meng. “Multi-distribution deep belief network for speech synthesis”. In: *2013 IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*. IEEE. 2013, pages 8012–8016.

- [57] Yao Qian et al. “On the training aspects of deep neural network (DNN) for parametric TTS synthesis”. In: *2014 IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pages 3829–3833.
- [58] Heiga Ze, Andrew Senior, and Mike Schuster. “Statistical parametric speech synthesis using deep neural networks”. In: *2013 IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*. IEEE. 2013, pages 7962–7966.
- [59] Heiga Zen and Andrew Senior. “Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis”. In: *2014 IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pages 3844–3848.
- [60] Yuchen Fan et al. “TTS synthesis with bidirectional LSTM based recurrent neural networks”. In: *Fifteenth Annual Conference of the International Speech Communication Association*. 2014, pages 1964–1968.
- [61] Yuxuan Wang et al. “Tacotron: A fully end-to-end text-to-speech synthesis model”. In: *arXiv preprint* (2017).
- [62] Aäron Van Den Oord et al. “Wavenet: A generative model for raw audio”. In: *CoRR abs/1609.03499* (2016).
- [63] Aaron van den Oord et al. “Parallel WaveNet: Fast high-fidelity speech synthesis”. In: *arXiv preprint arXiv:1711.10433* (2017).
- [64] Ye Jia et al. “Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis”. In: *arXiv preprint arXiv:1806.04558* (2018).

索引

A

- Acoustic Feature, 声学特征 34
Acoustic Model, 声学模型 13
Activation, 激活函数 42
Analog to Digital Conversion, 模数转换 .. 10
Attention, 注意力 17

B

- Beam Search, 集束搜索 42

C

- Cepstral Mean and Variance Normalization 34
Cepstrum, 倒谱 37
Classification, 分类 16
Classifier, 分类器 42
Computational Graph, 计算图 19, 41
Connectionist Temporal Classification..... 17
Convolution, 卷积 36
Convolutional Neural Network, 卷积神经网
络 17
Cross Entropy, 交叉熵 42

D

- Decision Tree, 决策树 15
Decoding, 解码 15
Deep Neural Network, 深度神经网络 16
Deterministic, 确定性 33
Disambiguation, 消歧 33
Discrete Cosine Transform, DCT, 离散余弦变
换 37

- Discrete Fourier Transform, 离散傅里叶变换
10

- Discriminative Model, 判别式模型 16
Dithering, 抖动 35
Dynamic Feature, 动态特征 38

E

- Emission Probability, 发射概率 15
Energy, 能量 38
Envelope, 包络 37

F

- Fast Fourier Transform, FFT, 快速傅里叶变
换 36

Feedforward Neural Network, 前向神经网络	16
Force Alignment	40
Formant, 共振峰	10
Frame, 帧	14
Fundamental Frequency, 基频	37

G

Gaussian Mixture Model, 高斯混合模型	16
Generative Model, 生成式模型	16

H

Hamming Window, 海明窗	35
Hanning Window, 汉宁窗	35
Harmonic, 谐波	37
Hidden Markov Model, 隐马尔可夫模型	14

I

Inverse Discrete Fourier Transform, 离散傅里叶反变换	37
--	----

L

Label, 标签	16
Language Model, 语言模型	13
Language Recognition, 语种识别	29
Likelihood, 似然	16
Linear Discriminant Analysis, 线性判别分析	40

M

Maximum Likelihood Estimation, 最大似然估计	40
Maximum Likelihood Linear Transform, 最大似然线性变换	40
Maximum Mutual Information, MMI	42
Mean Squared Error, 均方误差	42
Mel Filter Bank, 梅尔滤波器组	34, 37

Mel Frequency Cepstral Coefficient, 梅尔频率倒谱系数	34
--	----

Mel Frequency, 梅尔频率	36
---------------------	----

Mel scale	36
-----------	----

Minimum Phone Error, MPE	42
--------------------------	----

N

N-gram Grammar, n 元语法	17
-----------------------	----

Nyquist frequency, 奈奎斯特频率	36
---------------------------	----

O

Out Of Vocabulary, 集外词	33
------------------------	----

P

Perceptual Linear Prediction, PLP	35
Phone, 音素	15
Posterior Probability, 后验概率	16
Pre-emphasis, 预加重	35
Prior Probability, 先验概率	16
Probability Density Function, 概率密度函数	16

16

Q

Quantization Error, 量化误差	35
--------------------------	----

R

Rectangular Window, 矩形窗	35
Recurrent Neural Network, 循环神经网络	16
Reference, 推断	42
Representation Learning, 表征学习	10

S

Senones	15
Sequence-Discriminative Training, 序列判别训练	42
Speaker Recognition, 说话人识别	29
Spectral Leakage, 频谱泄漏	35
Spectral Tilt, 频谱倾斜	35

- Spectrogram, 频谱图 10, 36
Spectrum, 频谱 36
Speech Recognition, 语音识别 13
state Minimum Bayes Risk, sMBR 42
Supervised Learning, 有监督学习 16

T

- Time Delay Neural Network, 时延神经网络 42
Triangular Filter Bank, 三角滤波器组 37
Triphone, 三音素 15

V

- Vocal Cord, 声带 37
Vocal Tract, 声道 37
Voice Activity Detection, 语音活性检测 34

W

- Waveform, 波形图 10
Weight, 权重 42
Weighted Finite State Transducer, 加权有限状态转换器 16, 40
Windowing, 加窗 35
Word Error Rate, WER, 词错误率 40