第 3 章 语音识别原理与技术

【内容导读】语音识别是语音信息处理的重要研究方向和应用技术。基于模板匹配的语音识别技术是 20 世纪 70 年代的主流技术。上世纪 80 年代以来，语音识别算法逐步从模板匹配算法进展到基于统计模型的算法。2010 年之前，基于 GMM-HMM 是语音识别的经典声学模型。2010 年前后，深度学习算法在语音识别领域取得巨大成功，DNN-HMM 逐步替代了 GMM-HMM。2015 年前后，端到端模型受到越来越多的关注，已经成为学术研究的主要方法，并开始逐步应用到产业界。除了声学模型，本章也会介绍语音识别的另外两个主要组成部分：语言模型和解码算法。本章的最后，介绍基于 HTK、KALDI、ESPNET 等工具的语音识别实践，便于读者搭建主流语音识别系统。

3.1 语音识别概述

3.1.1 简介

语音识别（Automatic Speech Recognition, ASR）目的是实现语音到文本的转换，也称语音转文本（Speech To Text，STT），这样的表述可以与语音合成的文本转语音（Text To Speech，TTS）对应。语音识别是一门语言学、声学、统计学、计算机科学等多学科交叉的科学。近些年，随着数据、算力和算法的提升，语音识别性能取得了巨大飞跃，并得以大规模落地应用，越来越多的语音识别产品走进了人们生活，如手机语音助手、智能音箱、智能语音客服、语音会议转写系统等。

语音识别根据识别对象，可分为孤立词识别、关键词识别和连续语音识别三类。孤立词识别是识别出事先设定的孤立词，如语音操控手机时的“打开**”、“关机”等；关键词识别是在连续语音中检测指定关键词是否出现以及在何处出现，如手机语音助手唤醒中小米手机的“小爱同学”和华为手机的“小艺小艺”等；连续语音识别需要将连续语音转换成对应文本，一般情况下说语音识别，是指连续语音识别，其中根据是否连接网络，连续语音识别又可分为在线语音识别和离线语音识别。

按词表大小、说话人限制、语音内容、说话方式和识别场景，语音识别难度由简到难可做如下划分：从小词汇量、特定人、孤立词的识别，到大词汇量、非特定人、连续语音的识别；从特定领域的标准口语识别到非特定领域的自然口语的识别；从简单场景下（安静、近场）的识别到复杂场景下（多噪音、混响、远场）的识别等。

目前语音识别技术已经取得很大进步，但距全场景高准确率的最终语音识别目标还有较大差距，尽管如此我们对未来语音识别技术的发展仍充满信心。

3.1.2 发展历史

现代语音识别可以追溯到 1952 年，这一年贝尔实验室发布了可以识别 one 到 ten 的十个英文单词的机器，标志着现代语音识别的开始。从 20 世纪 50 年代到现在经过将近 70 年的发展，语音识别从开始的性能极不稳定的孤立词识别，到现在的多数场景下表现稳定的大词汇量连续语音识别（Large Vocabulary Continuous Speech Recognition, LVCSR）[1]。

早期语音识别主要集中在小词汇量和孤立词的识别。从 20 世纪 50 年代到 60 年代，孤立词识别主要使用基于语音学、语言学等规则的方法，包括元音部分的频谱特征、从带通滤波器组获得的频谱参数、声道的时变估计技术等特征和技术。从 20 世纪 70 年代到 80 年代前期，基于模板匹配是该阶段比较成功的方法。首先将孤立词提取出来的特征及对应文

本作为模板保存起来，当识别语音时，通过相同的方法提取待识别语音的特征，并与已有的特征模板逐个进行比较，选取相似度最高的特征模板对应的文本作为该语音识别的输出。模板匹配的经典方法是动态时间规整（Dynamic Timing Warping, DTW）算法。DTW 是一种非线性归正算法，能够较好计算输入语音与模板语音之间发音长度不匹配的问题，在当时是一种很成功的模板匹配算法。但是，DTW 方法稳定性差，在语音信号的变化性小的情况下识别效果比较好，在复杂的说话人特性以及声学环境变化场景下，语音识别的性能就急剧下降，而且此方法难以进行通用场景的大词汇量连续语音识别。

20 世纪 80 年代，基于统计模型的方法逐渐替代基于模板匹配的方法成为语音识别的主流方法。在神经网络模型兴起之前，基于统计模型的方法一直被高斯混合模型（Gaussian Mixture Model，GMM）和隐马尔可夫模型（Hidden Markov Model，HMM）组成的 GMM-HMM 声学模型所统治[2]。HMM 有效的解决了语音的长时时变和短时平稳的特点，每个建模单元使用 HMM 的 3-5 个状态来表示，根据语音的短时平稳特性，认为每个状态保持稳定，同时每个状态都有自己的观测概率，由 GMM 来输出，状态与状态之间有转移概率，实验时一般保持固定，这样音频特征序列到音素序列的转换被建模成概率问题。由于 GMM-HMM 模型可以使用大量语音数据来进行训练，从而能得到更好的参数，这使得模型的稳定性和识别准确率得到大幅度提升。

2006 年 Hinton 提出了深度置信网络[3]，激起了学术界对深度神经网络（Deep Neural Network，DNN）的研究，并于 2009 年提出 DNN-HMM 模型，通过使用 DNN 来代替 GMM 输出 HMM 各个状态的观测概率，在 TIMIT 数据集上取得当时最好的结果（State-of-The-Art，SOTA）[4]。2011 年微软研究院的研究者们以上下文相关（Context Dependent，CD）的三音子为建模单元，提出 CD-DNN-HMM 声学模型，首次将 DNN 应用于 LVCSR 任务，相比于 GMM-HMM，识别错误率下降 20%左右[5]。此后卷积神经网络（Convolutional Neural Network，CNN）和循环神经网络（Recurrent Neural Network, RNN）[5]及其变体长短时记忆网络（Long Short-Term Memory，LSTM）[6]和门控循环单元（Gated Recurrent Unit, GRU）[7]单个或多个和隐马尔可夫组成了 NN-HMM 模型。借助于神经网络强大的学习能力，语音识别稳定性和准确率达到新的高度。

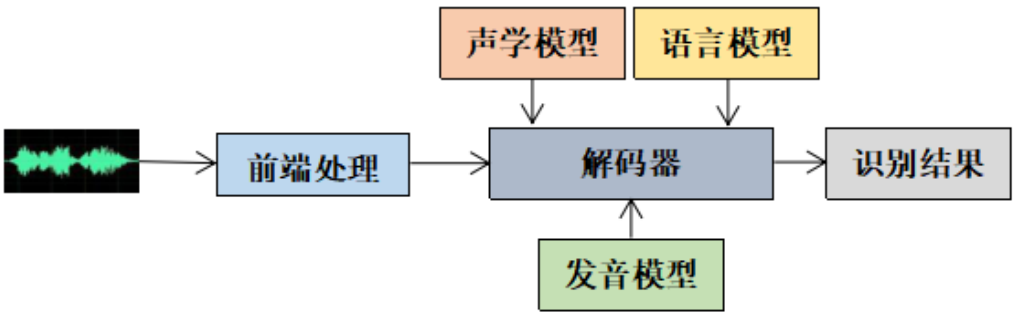


图 3-1 传统语音识别框架

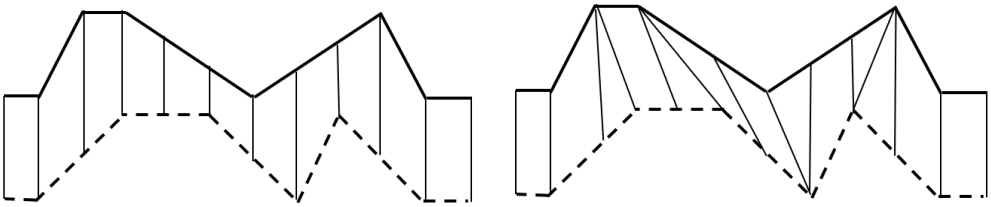
基于 HMM 的语音识别被称为传统语音识别（框架如图 3-1），其由声学模型、发音模型和语言模型组成，这些模型及其解码算法将在 3.2 节到 3.4 节介绍，其中各个模块分开优化，很难得到全局最优。2006 年 Alex Graves 等提出连接时序分类（Connectionist Temporal Classifier，CTC）算法[8]，并于 2014 年将其引入到语音识别领域[9]，接在神经网络模型的最后，解决了长的音频特征序列和短的单词序列的对齐问题，由于可以直接将音频特征转换成文本，因而被称作端到端语音识别。相比于传统语音识别，端到端语音识别（框架如图 3-2）还原了语音识别序列到序列的本质，将声学模型、发音模型和语言模型联合优化，极大地简化了语音识别流程，此后各种端到端模型层出不穷[10-12]，并极大推动了语音识

别的发展。虽然当前传统语音识别模型仍活跃于工业界，但随着端到端模型的深入研究，识别性能得以进一步提升，端到端语音识别会逐步替代传统语音识别。我们将在第八章详细介绍端到端语音识别模型。

图 3-2 端到端语音识别框架

3.1.3 基于模板匹配的语音识别

20 世纪 70 年代到 80 年代前期的语音识别主要采用模板匹配的方法，具体为训练时保存发音模板和对应内容到模板库，识别时再将所要识别的语音和模板库的发音模板逐一计算相似度，选择相似度最高的发音模板所对应的内容为识别结果。在计算相似度时，由于不同人有不同的语速，同一个词中不同音素发音速度不同，即使同一人不同时刻相同内容的发音也有所不同。尽管同一内容的发音整体相似，但时间上也不是完全对应，如果使得同一时刻上的点相互对应，如图 3-3（a）所示，相似度结果会比实际差很多。动态时间规整（Dynamic Time Warping，DTW）算法可以很好的计算两段不同长度时间序列的相似度，如图 3-3（b）所示，其可以使每个点尽可能找到和它最相似的点，这种情况下一个点可以对应多个点，同时多个点也可以对应一个点，这样计算相似度比直接计算会得到更准确的结果，同时也更为合理。下来我们来详细介绍 DTW。



（a）直接比较序列相似性　　（b）采用 DTW 比较序列相似性

图 3-3 直接比较与采用 DTW 算法比较序列相似性

首先有两个矢量序列$\mathbf{A}=\{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_\mathrm{I}\}$和$\mathbf{B}=\{\mathbf{b}_1,\mathbf{b}_2,\ldots,\mathbf{b}_\mathrm{J}\}$（以语音为例，$\mathbf{a}_\mathrm{i}$和$\mathbf{b}_\mathrm{j}$分别指第 i 帧和第 j 帧的特征向量），我们构造一个$\mathrm{I}\times\mathrm{J}$的矩阵$\mathbf{D}$，矩阵中的一个元素$\mathrm{D}(\mathrm{I},\mathrm{j})$表示$\mathbf{a}_\mathrm{i}$和$\mathbf{b}_\mathrm{j}$两点对齐计算的距离（距离越小相似度越大），一般采用欧氏距离，即$\mathrm{D}(\mathrm{I},\mathrm{j})=\mathrm{d}(\mathbf{a}_\mathrm{I},\mathbf{b}_\mathrm{j})=\sum_{m=1}^{M}\left(a_i^m-b_j^m\right)^2$其中$1\le\mathrm{i}\le\mathrm{I}$，$1\le\mathrm{j}\le\mathrm{J}$，M 表示矢量$\mathbf{a}_\mathrm{i}$和$\mathbf{b}_\mathrm{j}$的维度。我们将矩阵可以看成一个网格如图 3-4，DTW 的目的就是找到通过这个网格中若干格点的路径，路径通过的格点就是两个序列对齐的点，所有对齐的点计算距离的和称为规整路径距离，这条路径被称作规整路径。

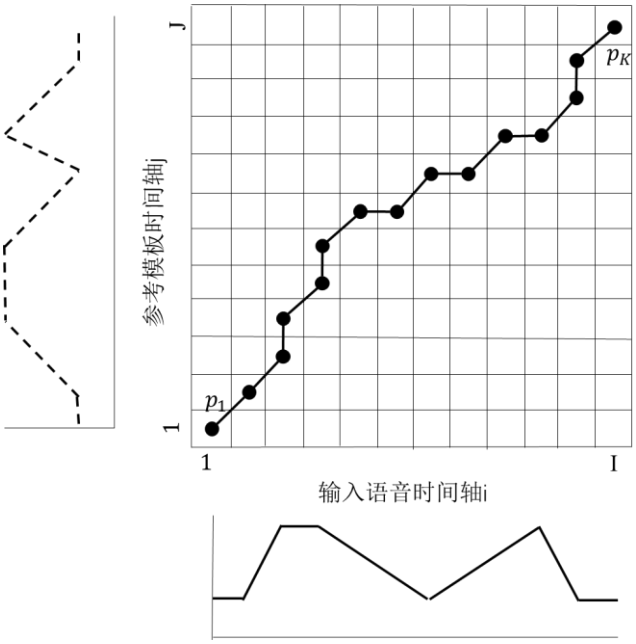


图 3-4 规整路径网格示例

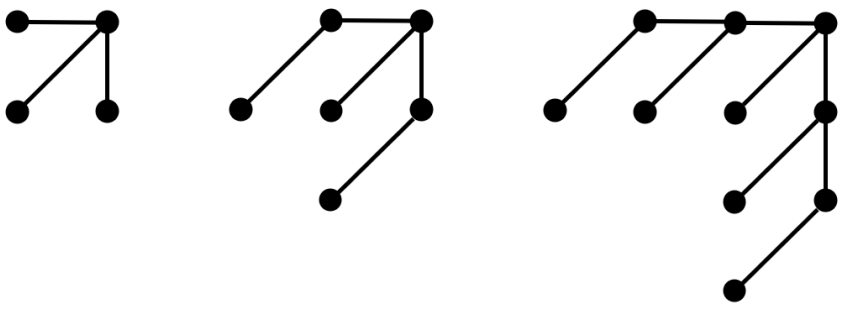
我们用P来表示该路径，具体形式为$P = p_1, p_2, \dots, p_K$，其中$\max(I, J) \le K \le I + J$，$p_k = (I, j)$表示路径经过坐标为$(I, j)$的格点，其中$1 \le k \le K$。这条路径需要满足如下几个约束条件：

（1） **边界条件：**必须保证$p_1 = (1,1)$，$p_K = (I, J)$，即所选路径必须从网格的左下角出发，到右上角结束，即两段序列的首和尾必须对齐。

（2） **连续性：**如果$p_k = (x_k, y_k)$，$p_{k+1} = (x_{k+1}, y_{k+1})$，必须保证$x_{k+1} - x_k \le 1$和$y_{k+1} - y_k \le 1$，即当前点只能和自己相邻的点对齐，这是为了确保序列 **A** 和 **B** 的每一点都有与之对齐的点。

（3） **单调性：**必须保证$x_{k+1} - x_k \ge 0$和$y_{k+1} - y_k \ge 0$，即保证路径随着时间单调向前进行。

（4） **最大规整量限制：**为了避免出现极端情况，规整路径所处区域需要进行限制，如规整路径必须位于某个平行四边形内。因此需要对局部路径进行约束，如图3-5，展示了三种典型的局部路径约束。



（a）受前一步约束　（b）受前两步约束　（c）受前三步约束

图 3-5 三种典型的局部路径约束

最终所求的路径必须为满足以上四个条件后的最短路径。 这里定义最小累计距离函数R(i,j)，表示到匹配点（i,j）为止的所有可能路径中的最佳路径的累计匹配距离。以图3-4的区域约束和图 3-5(a)的局部路径约束为例，DTW 的算法归纳如下：

（1）初始条件：$R(1,1) = D(1,1)W(2)$

（2）递推：对于当前路径的累加距离，采用如下递推公式，

$$R(i,j) = \min\begin{Bmatrix} R(i-1,j) + D(i,j)W(1) \\ R(i-1,j-1) + D(i,j)W(2) \\ R(i,j-1) + D(i,j)W(3) \end{Bmatrix} \quad (3.1)$$

其中在公式（3. 1）中，对于图 3-5(a)的局部路径约束，一般取距离加权指为 $W(1) = W(3) = 1$，$W(2) = 2$。

（3）最佳路径的匹配距离：最终$\frac{R(I,J)}{I+J}$为两个序列的匹配距离。

3.2 声学模型

声学模型是语音识别系统的重要组成部分，用于语音信号与声学单元之间的建模，是计算声学特征与声学模型之间的相似度。本节首先介绍马尔可夫链的基本概念。然后将概念扩展到隐马尔可夫模型上。隐马尔可夫模型在传统的语音识别系统中发挥了重要作用，成为了现在语音识别声学模型的主流方法。本节将详细介绍隐马尔可夫模型的三个基本算法： 前向-后向算法、Viterbi 算法、Baum-Welch 算法。混合高斯模型作为基于傅里叶频谱语音特征的统计模型，在语音特征建模中具有一定的优势，本节将详细介绍混合高斯模型及其对语音特征的建模方法。之后介绍如何将隐马尔可夫模型应用于语音识别系统中，与高斯混合模型共同进行声学模型建模。随着深度学习的发展，神经网络在语音声学建模中取得了成功，最后本节将介绍基于深度学习的声学模型。

在介绍声学模型之前，需要读者们先了解声学模型的基本建模单元以及如何选择建模单元。声学模型的建模是用来描述声学的基本单位。建模单元的选择是声学模型建模中基本而重要的问题，在实际应用中需要根据不同的需求选择不同的基元。在汉语连续语音识别中，可选择的建模单元包括词（word）、字（character）、音节（syllable）、半音节（semi-syllable）、声韵母（initial/final）和音素（phone）等。而在英文连续语音识别中，可选的建模单元包括词（word）、音素（phone）、元辅音（vowel/consonant）等。声学建模单元的选择一般基于语音学知识，也可以基于数据驱动的方式产生。使用数据驱动方式确定的基元，可能在语音学上没有什么明确的意义，但可以达到很好的性能。

在小词汇量的语音识别系统中，把词作为一个基本的语音单元建立模型，对于简化系统结构和训练过程是很有效的，因为连续语音识别中词与词之间的相互影响比词内的音素或音节的相互影响要小得多。但是，对于大词汇量的语音识别系统，采用词作为建模单元，词间的各种音联关系（如，“你好”一词中音素的关系）可能得不到充分的训练容易造成过拟合，并且需要大量的存储和复杂的计算。一般来说，声学建模单元越小，其数量也就越少，训练模型的工作量也越小；但是另一方面，单元越小，对上下文的敏感性越大，越容易受到前后相邻的影响而产生变异。

我们知道，发音时产生的音与其上下文有密切关系，因此在声学建模中，我们一般考虑上下文相关信息，这样声学建模单元就会变成上下文相关的单元。在基于隐马尔可夫的声学模型建模中，为考虑其上下文相关的信息，通常将一个建模单元分为三个，分别表示

一个单元的“前-中-后”的状态，如“三音素”是将一个上下文相关的音素表示为三个不同的状态。例如，三个单音素/t/, /iy/, /n/可表示为三音素/t-iy+n/，即由三个单音素组成。三音素/t-iy+n/与单音素/iy/类似，不同的是三音素/t-iy+n/考虑了音素/iy/的上下文，其中/t/是/iy/的上文，/n/是/iy/的下文。

下面，我们将详细介绍基于隐马尔可夫的声学模型以及基于深度学习的声学模型。

3.2.1 隐马尔可夫模型基本原理与算法

3.2.1.1 马尔可夫链

马尔可夫链（Markov chain）[13]是一种离散状态的马尔可夫序列，也是一般性的马尔可夫序列，为状态空间中经过从一个状态到另一个状态的转换的随机过程。该过程要求具备“无记忆”的性质，即下一状态的概率分布只能由当前状态决定，在时间序列中它前面的事件均与之无关。该性质也被称为马尔可夫性。除此之外，其状态空间还具有离散性和有限性[14]：$q_t \in \{s_j,\ \mathrm{j} = 1,2,\dots,\mathrm{N}\}$。其中每一个离散值都与状态空间集合中的一个状态$s_j$相关，$j$为索引。由于$s_j$与$j$一一对应，本书可以同等使用这两者。

具体地，一个一阶马尔可夫链$\mathbf{Q} = \mathrm{q}_1, \mathrm{q}_2, \dots, \mathrm{q}_\mathrm{T}$满足如下关系

$$\mathrm{P}\left(\mathrm{q}_\mathrm{t} = s_j \middle| \mathrm{q}_{\mathrm{t}-1} = s_i, \mathrm{q}_{\mathrm{t}-2} = s_k \dots\right) = \mathrm{P}\left(\mathrm{q}_\mathrm{t} = s_j \middle| \mathrm{q}_{\mathrm{t}-1} = s_i\right), \quad t = 1,2,\dots \mathrm{T} \tag{3.2}$$

其可被转移概率完全表示，定义为

$$\mathrm{P}\left(\mathrm{q}_\mathrm{t} = s_j \middle| \mathrm{q}_{\mathrm{t}-1} = \mathrm{s}_i\ \right) = \mathrm{a}_{\mathrm{ij}}(\mathrm{t}), \quad \mathrm{I}, \mathrm{j} = 1,2,\dots,\mathrm{N} \tag{3.3}$$

$\mathrm{a}_{\mathrm{i,j}}(\mathrm{t})$表示在 t 时刻从状态i转移到状态j的概率。如果这些转移概率与时间t无关，则得到齐次马尔可夫链。

（齐次）马尔可夫链的转移概率通常能方便地表示为矩阵形式：

$$\mathbf{A} = \left[\mathrm{a}_{\mathrm{i,j}}\right], \quad \text{其中，} \mathrm{a}_{\mathrm{ij}} \geq 0\ \forall \mathrm{i}, \mathrm{j};\ \sum_{\mathrm{j}=1}^{\mathrm{N}} \mathrm{a}_{\mathrm{I,j}} = 1\ \forall \mathrm{I} \tag{3.4}$$

A为马尔可夫链的状态转移矩阵。

给定马尔可夫链的转移概率，及t时间在状态 j 的输出概率$\mathrm{p}_\mathrm{j}(\mathrm{t}) = \mathrm{P}\left[\mathrm{q}_\mathrm{t} = \mathrm{s}_j\right]$，则根据下式可计算得到其在$\mathrm{t}+1$时间在状态 i 的输出概率，该计算是递归的。

$$\mathrm{P}_\mathrm{i}(\mathrm{t}+1) = \sum_{\mathrm{j}=1}^{\mathrm{N}} \mathrm{a}_{\mathrm{ji}}\, \mathrm{p}_\mathrm{j}(\mathrm{t}), \ \forall \mathrm{i} \tag{3.5}$$

如果一个马尔可夫链在状态转移矩阵A的作用下渐进收敛（达到平稳状态）：

$\lim_{\mathrm{t}\to\infty} \mathrm{p}_\mathrm{i}(\mathrm{t}) \to \pi(\mathrm{q}_i)$，其中，$\pi(\mathrm{q}_i)$表示在状态转移矩阵A的作用下离散值$\mathrm{q}_\mathrm{t}$从初始状态转移到状态i的概率。我们称$\mathrm{p}(\mathrm{s}_i)$为马尔科夫链的一个稳态分布。对于有稳态分布的马尔可夫

链来说，它的转移概率必须满足：

$$\bar{\pi}(s_i) = \sum_{j=1}^{N} a_{ji}\bar{\pi}(s_j), \forall i \qquad (3.6)$$

$\bar{\pi}(s_i)$ 表示有稳态分布的马尔科夫链从初始状态转移到状态 i 的概率。

3.2.1.2 **隐马尔可夫模型**

前文讨论的马尔可夫链可以看作一段能够生成可观测输出的序列。因为它的输出的观测和每一个状态是一一对应的，即给定一个状态唯一对应一种观测或事件，没有任何随机性。所以，它又可称为可观测的马尔可夫序列。这种没有任何随机性的缺陷，导致其在描述现实世界中的信息时过于局限。

隐马尔可夫作为马尔科夫链的一种扩展，在每个状态中引入了随机性。在马尔可夫模型中，由于其每一个确定的观测值或事件都有唯一一个状态与之对应，因此状态对于观察者来说是直接可见的，状态的转换概率便是模型全部的参数。而在隐马尔可夫模型（图 3-6）中，是用一个观测的概率分布与每一个状态对应，而不是一个确定的观测值或事件，这就在马尔可夫序列的状态中引入了随机性，使其状态并不能被直接观测。要注意的是，各个状态的观测概率不能没有任何重合，否则在某个固定范围内的观测值总能找到唯一一个状态与之对应，这不符合隐马尔可夫模型状态不可观测的特点。

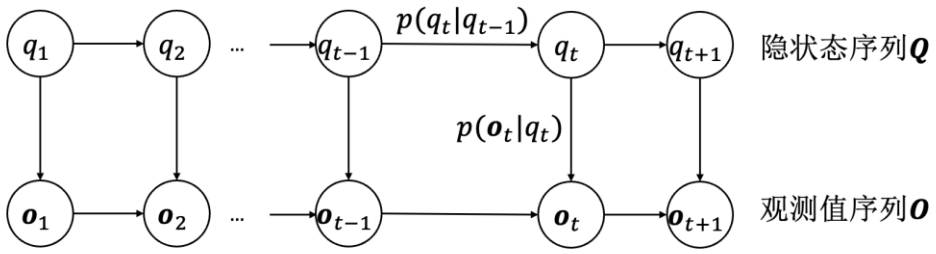


图 3-6 隐马尔可夫模型图

当隐马尔可夫序列被用来描述现实世界的信息时，比如用于拟合这种信息的统计模型，我们称为隐马尔可夫模型（HMM）。自 20 世纪 80 年代以来，HMM 被应用于语音识别，取得重大成功。随后，HMM 也被成功应用到语音处理领域的其他方向，如语音合成、语音增强等。HMM 可以描述语音信号中不平稳但有规律的空间变量，将其应用于语音声学模型，可以分段的处理短时平稳的语音特征，并学习拟合到全局语音特征。在后面的部分，我们将详细介绍 HMM 的定义以及一些基本算法。这些算法主要应用于语音声学模型的训练和预测中。

一、定义

隐马尔可夫模型[15]是马尔可夫链的一种，它的状态不能直接观察到，但能通过观测向量序列观察到，每一个观测向量都是通过某些概率密度分布与每一个状态对应，每一个观测向量是由一个具有相应概率密度分布的状态序列产生。所以，隐马尔可夫模型是一个双重随机过程。

二、模型表示

HMM 可以用下列参数来描述，包括状态集合、观测值集合和 3 个概率矩阵：

1）状态集合：$\mathbf{S}=\{s_j,\ j=1,2,\dots,N\}$

N：状态数目。这些状态之间满足马尔可夫性质，是马尔可夫模型中实际所隐含的状态。这些状态不可被直接观测而得到。

2）可观测值集合：$\mathbf{V}=\{\boldsymbol{v}_1,\boldsymbol{v}_2,\dots\boldsymbol{v}_K\}$

K：每个状态对应的可能的观察值数目。在模型中与隐含状态相关联，可通过直接观测而得到。时刻 *t* 观察到的观察值为$\mathbf{o}_t$, $\boldsymbol{o}_t\in\{\boldsymbol{v}_1,\boldsymbol{v}_2,\dots\boldsymbol{v}_K\}$。

3）初始状态概率矩阵：$\pi=[\pi_1\ \pi_2\ \dots\ \pi_N]$

表示隐含状态在初始时刻 t=1 的概率矩阵。其中$\pi_i=P(q_1=i)$。

4）状态转移概率矩阵：$\mathbf{A}=[a_{i,j}],\ 1\le i,j\le N$

描述了 HMM 中各个状态之间的转移概率，$a_{I,j}$表示在 t-1 时刻状态为s_i，且在 t 时刻状态为s_j的概率。（定义同马尔科夫链的状态转移矩阵）

5）观测值概率矩阵：$\mathbf{B}=[b_j(k)],\ 1\le j\le N, 1\le k\le K$

表示在 t 时刻、隐含状态是s_j的条件下，观察值为v_k的概率。

具体地，若观察概率分布为$P(\mathbf{o}_t|s_j), j=1,2,\dots,N$，且$\mathbf{o}_t$是离散的，表示在状态下$s_j$观测到的概率分布， 每个状态对应的概率分布用来表述观测序列 $\mathbf{V}=\{\mathbf{v}_1,\mathbf{v}_2,\dots\mathbf{v}_K\}$的概率：

$$b_j(\mathbf{o}_t)=P(\mathbf{o}_t=v_k|q_t=j),\ k=1,2,\dots K, j=1,2,\dots,N \tag{3.7}$$

若观察概率密度是连续的，那么概率密度函数中的参数Λ_j就可以表示 HMM 中s_j状态的特性。

有了这些参数之后，一般地，我们可以用$\lambda=(\mathbf{A},\mathbf{B},\pi)$三元组来简洁的表示一个隐马尔可夫模型。隐马尔可夫模型相比于标准马尔可夫模型，添加了可观测状态集合和这些状态与隐含状态之间的概率关系。

三、基本问题及其解决方法

前面介绍了 HMM 模型的定义和性质，下面我们来讨论 HMM 的三个基本问题（模型概率计算问题、解码问题、参数估计问题）及其解决方法[16-18]。

1．模型概率计算问题及算法

（a）问题描述

给定观测序列$\mathbf{O}=\{\mathbf{o}_1,\mathbf{o}_2,\dots,\mathbf{o}_T\}$和模型参数$\lambda=(\mathbf{A},\mathbf{B},\pi)$，怎样有效计算某一观测序列的概率$P(\mathbf{O}|\lambda)$，进而可对该 HMM 做出相关评估。例如，已有一些模型参数各异的 HMM，给定观测序列$\mathbf{O}=\{\mathbf{o}_1,\mathbf{o}_2,\dots,\mathbf{o}_T\}$，我们想知道哪个 HMM 模型最可能生成该观测序列。通常我们利用前向、后向算法分别计算每个 HMM 产生给定观测序列$\mathbf{O}$的概率，然后从中选出最优的 HMM 模型。

（b）直接计算法[16]

为了解决评估问题，即给定观测序列$\mathbf{O}=\{\mathbf{o}_1,\mathbf{o}_2,\dots,\mathbf{o}_T\}$和模型参数$\lambda=(\mathbf{A},\mathbf{B},\pi)$，计算某一观测序列的概率$P(\mathbf{O}|\lambda)$。最直接且最容易想到的方法就是直接计算法。具体算法如下：

由已知条件可计算，状态序列$\mathbf{Q}=\{q_1,q_2,\dots q_T\}$的概率是

$$P(\mathbf{Q}|\lambda)=\pi_{q_1}a_{q_1q_2}a_{q_2q_3}\dots a_{q_{T-1}q_T} \tag{3.8}$$

对固定的状态序列$\mathbf{Q}=\{q_1,q_2,\dots q_T\}$，观测序列$\mathbf{O}=\{\mathbf{o}_1,\mathbf{o}_2,\dots,\mathbf{o}_T\}$的概率是

$$P(\mathbf{O}|\mathbf{Q},\lambda)=b_{q_1}(\mathbf{o}_1)b_{q_2}(\mathbf{o}_2)\dots b_{q_T}(\mathbf{o}_T) \tag{3.9}$$

则$\mathbf{O}$和$\mathbf{Q}$同时出现的联合概率为

$$P(\mathbf{O},\mathbf{Q}|\lambda)=P(\mathbf{O}|\mathbf{Q},\lambda)P(\mathbf{Q}|\lambda)= \pi_{q_1}b_{q_1}(\mathbf{o}_1)a_{q_1q_2}b_{q_2}(\mathbf{o}_2)a_{q_2q_3}\dots a_{q_{T-1}q_T}b_{q_T}(\mathbf{o}_T) \tag{3.10}$$

对所有可能的状态序列$\mathbf{Q}$求和，得到观测序列$\mathbf{O}$的概率$P(\mathbf{O}|\lambda)$

$$P(\mathbf{O}|\lambda)=\sum_{\forall\mathbf{Q}}P(\mathbf{O}|\mathbf{Q},\lambda)P(\mathbf{Q}|\lambda)= \sum_{q_1,q_2\dots,q_T}\pi_{q_1}b_{q_1}(\mathbf{o}_1)a_{q_1q_2}b_{q_2}(\mathbf{o}_2)a_{q_2q_3}\dots a_{q_{T-1}q_T}b_{q_T}(\mathbf{o}_T) \tag{3.11}$$

这种方法简单，但计算的时间复杂度为$O(TN^T)$，当时间序列过长时，所需要的计算时间是巨大的，几乎是不可计算的。因此提出了前向算法来简化计算。

（c）前向算法 (Forward Algorithm) [14]

如图 3-7，给定隐马尔可夫模型参数$\lambda=(\mathbf{A},\mathbf{B},\pi)$，定义到时刻 *t* 部分观测序列为$\mathbf{o}_1,\mathbf{o}_2,\dots,\mathbf{o}_t$，且状态为$s_i$（为方便表示，在后面的算法中状态$s_i$简记为 i）的前向概率为

$$\alpha_t(i)=P(\mathbf{o}_1,\mathbf{o}_2,\dots,\mathbf{o}_t,q_t=i|\lambda) \tag{3.12}$$

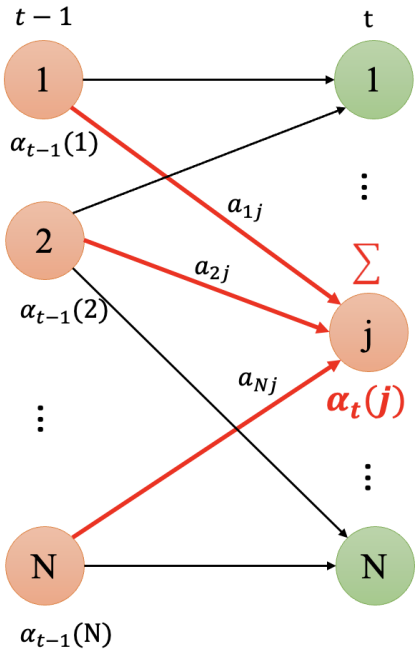


图 3-7 前向概率计算示意图

首先，设置前向概率的初值

$$\begin{aligned}\alpha_1(\mathrm{i}) &= \mathrm{P}(\mathbf{o}_1, q_1 = \mathrm{i}|\lambda)\\ &= \mathrm{P}(q_1 = \mathrm{i})\mathrm{P}(\mathbf{o}_1|q_1 = \mathrm{i}, \lambda)\\ &= \pi_j b_i(\mathbf{o}_1),\\ &\mathrm{i} = 1,2,\dots,\mathrm{N}\end{aligned} \tag{3.13}$$

然后，由初值递推出每时间步的前向概率，对 t= 2,3, … , T，j=1,2,…,N

$$\begin{aligned}\alpha_t(\mathrm{j}) &= \mathrm{P}(q_t = \mathrm{j}, \mathbf{o}_t|\lambda)\\ &= \sum_{i=1}^{N} \mathrm{P}(q_{t-1} = \mathrm{i}, q_t = \mathrm{j}, \mathbf{o}_{t-1}, \mathbf{o}_t|\lambda)\\ &= \sum_{i=1}^{N} \mathrm{P}(q_{t-1} = \mathrm{i}, \mathbf{o}_{t-1}|\lambda)\, \mathrm{P}(q_t = \mathrm{j}, \mathbf{o}_t|\lambda)\\ &= \sum_{i=1}^{N} \mathrm{P}(q_t = \mathrm{j}, \mathbf{o}_t|\lambda)\, \alpha_{t-1}(\mathrm{j})\\ &= \sum_{i=1}^{N} \mathrm{P}(\mathbf{o}_t|q_t = \mathrm{j}, q_{t-1} = \mathrm{i}, \lambda)\, \mathrm{P}(q_t = \mathrm{j}|q_{t-1} = \mathrm{i}, \lambda)\alpha_{t-1}(\mathrm{j})\\ &= \left[\sum_{i=1}^{N} \alpha_{t-1}(\mathrm{i})a_{ij}\right] b_j(\mathbf{o}_t),\ \mathrm{j} = 1,2,\dots \mathrm{N}\end{aligned} \tag{3.14}$$

最后，计算某一观测序列的概率

$$\mathrm{P}(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \mathrm{P}(\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T, q_t = \mathrm{i}|\lambda) = \sum_{i=1}^{N} \alpha_T(\mathrm{i}) \tag{3.15}$$

这种算法的时间复杂度是$O(TN^2)$，相比于直接计算法，直接引用了前一时刻的计算结果，避免重复计算，使计算效率大大提高。

（d）后向算法 (Backward Alogorithm)

后向算法与前向算法相反，如图 3-8，给定隐马尔可夫模型参数$\lambda = (\mathbf{A}, \mathbf{B}, \pi)$， 定义从时刻 *t+1* 开始时刻 *T* 为止部分观测序列为$\mathbf{o_{t+1}}, \mathbf{o_{t+2}}, \dots, \mathbf{o_T}$，且状态为 *i* 的后向概率为

$$\beta_t(i) = P(\mathbf{o_{t+1}}, \mathbf{o_{t+2}}, \dots, \mathbf{o_T} | q_t = i, \lambda) \tag{3.16}$$

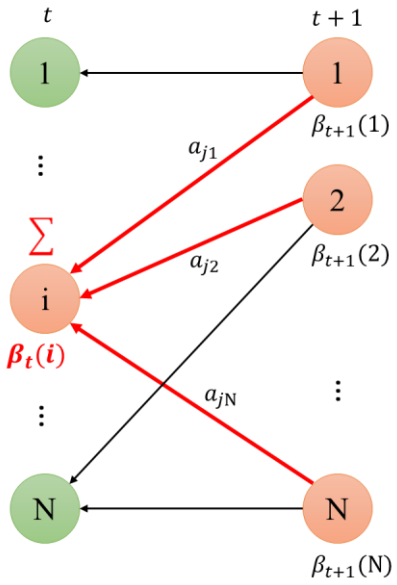


图 3-8 后向概率计算示意图

首先，设置后向概率的初值：

$$\beta_T(i) = 1, i = 1,2, \dots, N \tag{3.17}$$

对$t = T-1, \dots, 1$, i=1,2,…,N

$$
\begin{aligned}
\beta_t(i) &= P(\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, \dots, \mathbf{o}_T | q_t = i, \lambda) \\
&= \frac{P(\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, \dots, \mathbf{o}_T, q_t = i, \lambda)}{P(q_t = i, \lambda)} \\
&= \frac{\sum_{j=1}^{N} P(\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, \dots, \mathbf{o}_T | q_t = i, q_{t+1} = j, \lambda) P(q_t = i, q_{t+1} = j, \lambda)}{P(q_t = i, \lambda)} \\
&= \sum_{j=1}^{N} P(\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, \dots, \mathbf{o}_T | q_{t+1} = j, \lambda) \frac{P(q_t = i, q_{t+1} = j, \lambda)}{P(q_t = i, \lambda)} \\
&= \sum_{j=1}^{N} P(\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, \dots, \mathbf{o}_T | q_{t+1} = j, \lambda) a_{ij} \\
&= \sum_{j=1}^{N} P(\mathbf{o}_{t+2}, \mathbf{o}_{t+3}, \dots, \mathbf{o}_T | q_{t+2} = j, \lambda) P(\mathbf{o}_{t+1} | q_{t+1} = j, \lambda) a_{ij} \\
&= \sum_{j=1}^{N} \beta_{t+1}(j) b_j(\mathbf{o}_{t+1}) a_{ij} \qquad (3.18)
\end{aligned}
$$

最后，计算某一观测序列的概率

$$
P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \pi_i b_i(\mathbf{o}_1) \beta_1(i) \qquad (3.19)
$$

2．模型解码问题及算法

（a）问题描述

给定观测序列$\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T\}$和模型参数$\lambda = (\mathbf{A}, \mathbf{B}, \pi)$，怎样寻找某种意义上最优的隐状态序列$\boldsymbol{Q}^* = \{q_1^*, q_2^*, \dots, q_T^*\}$，使得$P(\boldsymbol{O}, \boldsymbol{Q}^*|\lambda)$取得最大值。在这类问题中，我们感兴趣的是隐马尔可夫模型中隐含状态，这些状态不能直接观测但却更具有价值，通常利用 Viterbi 算法来寻找。

（b）维特比算法 (Viterbi Algorithm) [17]

HMM 通过 Viterbi（维特比）算法解决解码问题。维特比算法是一个特殊但应用最广的动态规划算法。动态规划是一种分而治之地解决复杂问题的方法，它通过将复杂问题分解成一些更简单的问题来实现目标。利用动态规划，可以解决任何一个图中的最短路径问题。在 HMM 中，一条路径对应一个状态序列。最优路径具有以下特性：如果最优路径在时刻 *t* 通过结点 q_t^*，那么这一路径从结点q_t^*到终点q_T^*的部分路径，对于从q_t^*到q_T^*的所有可能的部分路径来说，必须是最优的。Viterbi 算法根据这一特性，将全局的最优状态序列动态地分解成到每个时间步为止的最优路径。具体过程如下：

首先，引入两个相关的变量δ和ψ。

如图 3-9，定义在 *t* 时刻状态为 *i* 的所有单个路径$(q_1, q_2, \dots, q_t)$中概率最大值为

$$
\delta_t(i) = \max_{q_1, q_2, \dots, q_t} P(q_t = i, q_1, q_2, \dots, q_{t-1}, \mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_t | \lambda), i = 1,2, \dots N \qquad (3.20)
$$

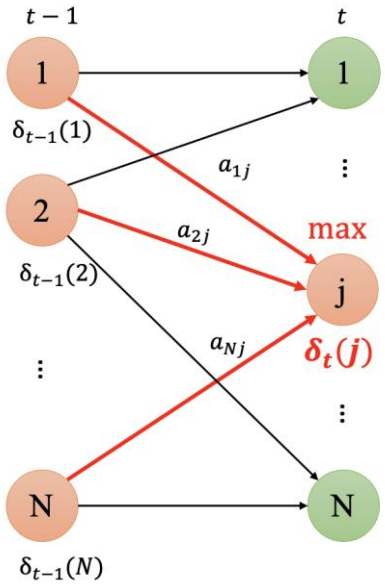


图 3-9 $\delta_t(i)$计算示意图

(3.20)计算$\delta_t(i)$的递推公式为，对于$t = 2,3,\dots,T$

$$\begin{aligned}\delta_t(j) &= \max_{q_1,q_2,\dots,q_t} P(q_t = j, q_1, q_2, \dots, q_{t-1}, \mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_t|\lambda) \\ &= \max_{1\le i\le N}\left[\delta_{t-1}(i)a_{ij}\right]b_j(\mathbf{o}_t)\end{aligned} \tag{3.21}$$

根据递推公式可计算出$\delta_T(i)$，即为所求的最优状态序列的概率。下面介绍如何找出最优状态路径。

如图 3-10，定义在 *t* 时刻的状态为 *j* 的所有单个路径$(q_1, q_2, \dots, q_t)$中概率最大的路径的第$t-1$个结点为

$$\varphi_t(j) = \arg\max_{1\le i\le N}\left[\delta_{t-1}(i)a_{ij}\right],\ j = 1,2,\dots,N \tag{3.22}$$

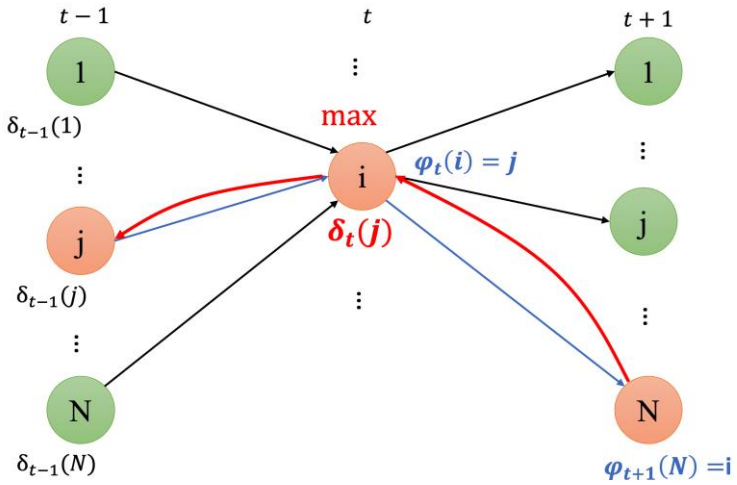


图 3-10 $\varphi_t(i)$计算示意图

有了两个变量关于概率和路径的变量，我们可以根据已知的模型参数$\lambda = (\mathbf{A}, \mathbf{B}, \pi)$和观测$\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T\}$，计算出最优路径$\mathbf{Q}^* = \{q_1^*, q_2^*, \dots, q_T^*\}$。

首先，初始化两个变量

$$\delta_1(i) = \pi_i b_i(\mathbf{o}_1),\ i = 1,2,\dots,N \tag{3.23}$$

$$\varphi_1(i) = 0,\ i = 1,2,\dots,N \tag{3.24}$$

对$t = 2,3,\dots,T$

$$\delta_t(j) = \max_{1\le i\le N}\left[\delta_{t-1}(i)a_{ij}\right]b_j(\mathbf{o}_t),\ j = 1,2,\dots,N \tag{3.25}$$

$$\varphi_t(j) = \arg\max_{1\le i\le N}\left[\delta_{t-1}(i)a_{ij}\right],\ j = 1,2,\dots,N \tag{3.26}$$

最大概率和最优路径的最后一个结点为

$$P^* = \max_{1\le i\le N}\delta_T(i) \tag{3.27}$$

$$q_T^* = \arg\max_{1\le i\le N}[\delta_T(i)] \tag{3.28}$$

最优路径回溯。对$t = T-1, T-2, \dots 1$

$$q_t^* = \varphi_{t+1}(q_{t+1}^*) \tag{3.29}$$

求得最优路径$\mathbf{Q}^* = \{q_1^*, q_2^*, \dots, q_T^*\}$。

3．模型参数估计问题及算法

（a）问题描述

HMM的模型参数$\lambda = (A, B, \pi)$未知，如何调整这些参数以使观测序列$O = \{o_1, o_2, \dots, o_T\}$的概率$P(O|\lambda)$尽可能的大。通常使用 Baum-Welch 算法解决。

在语音识别的隐马尔可夫模型中，每个建模单元（单词/音素/字等等）生成一个对应的 HMM，每个观测序列由一个建模单元的语音构成，每个建模单元一般由多个状态相关，这些状态在识别中是不可被直接观测到的。在模型训练过程中，通过迭代调整模型参数，使得模型最好地解释（例如概率最大）已知观测序列（学习过程）。建模单元的识别是通过评估进而选出最有可能产生观测序列所代表的读音的 HMM 而实现的（评估问题）。通过 Viterbi 算法来寻找在给定观测序列的情况下最后可能的隐含状态序列，为识别提供信息（解码问题）。

（b）Baum-Welch 算法[19]

在 HMM 中，用 Baum-Welch 算法来学习参数，也就是期望最大化（Expectation-Maximization，EM）算法的一个特例。EM 算法是一种通用的用于解决最大化似然估计的迭代算法。每次迭代分为两步，步骤 E（期望）估计状态占用概率，M 步基于估计的状态占用概率，重新估计模型参数$\lambda = (\mathbf{A}, \mathbf{B}, \pi)$。

在语音声学模型中，HMM 通常用混合高斯模型（Gaussian Mixture Model，GMM）来描述观测概率。也就是观测概率$\mathbf{o}_i \sim N(\boldsymbol{\mu} \in R^D, \boldsymbol{\Sigma} \in R^{D\times D})$，$i = 1,2, \dots T$。这里以 GMM-HMM 模型为例来介绍 Baum-Welch 算法（EM 算法）。GMM 将在后面的小节中介绍，可以先跳过这节，学完 3.2.2 小节的 GMM 以后再来仔细阅读。

给定模型$\lambda = (\mathbf{A}, \mathbf{B}, \pi)$和观测$\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T\}$，定义在 *t* 时刻处于状态 *i* 的概率，即为状态占用概率

$$\gamma_t(i) = P(q_t = i|\mathbf{O}, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^N \alpha_T(I)} \tag{3.30}$$

在 *t* 时刻处于状态 *i* 且在$t+1$时刻处于状态 *j* 的概率为

$$\begin{aligned} \xi_t(i,j) &= P(q_t = i, q_{t+1} = j|\mathbf{O}, \lambda) \\ &= \frac{P(q_t = i, q_{t+1} = j, \mathbf{O}|\lambda)}{P(\mathbf{O}|\lambda)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^N \alpha_T(i)} \end{aligned} \tag{3.31}$$

且有$\gamma_t(i) = \sum_{k=1}^N \xi_t(i,k)$。在 E 步计算出这两个概率，在 M 步更新以下参数：

$$\hat{\mu}_{jk}=\frac{\sum_{t=1}^{T}\xi_t(j,k)\mathbf{o}^t}{\sum_{t=1}^{T}\xi_t(j,k)} \tag{3.32}$$

$$\hat{\Sigma}_{jk}=\frac{\sum_{t=1}^{T}\xi_t(j,k)(\mathbf{o}_t-\hat{\mu}_{jk})(\mathbf{o}_t-\hat{\mu}_{jk})^{T}}{\sum_{t=1}^{T}\xi_t(j,k)} \tag{3.33}$$

$$\hat{c}_{jk}=\frac{\sum_{t=1}^{T}\xi_t(j,k)}{\sum_{t=1}^{T}\sum_{k}\xi_t(j,k)} \tag{3.34}$$

$$\hat{\alpha}_{ij}=\frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\sum_{k=1}^{N}\xi_t(i,k)}=\frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \tag{3.35}$$

$$\hat{\pi}_i=\gamma_1(i) \tag{3.36}$$

一直重复 E 步和 M 步直至收敛。

3.2.2 基于 GMM-HMM 的声学模型及语音识别技术

一、单高斯模型

高斯分布即正态分布，是最常见概率分布模型，它经常被用来刻画一些随机量的变化情况。由于正态分布函数反映了自然界中普遍存在的有关变化量的一种统计规律，以及其非常好的数学性质而被广泛应用于语音识别。

当样本数据x是一维数据时，高斯分布遵从下方概率密度函数（Probability Density Function，PDF）[20]：

$$P(x|\theta)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \tag{3.37}$$

其中，μ为 x 的均值（期望），σ为标准方差。

当样本数据 $\mathbf{x}=(x_1,x_2,\dots,x_D)^T$为多维数据时，遵从下方概率密度函数[20]:

$$P(\mathbf{x}|\theta)=\frac{1}{(2\pi)^{\frac{D}{2}}(\boldsymbol{\Sigma})^{\frac{1}{2}}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \tag{3.38}$$

其中，$\boldsymbol{\mu}=(\mu_1,\mu_2,\dots,\mu_D)^T=E(\mathbf{x})$，$\boldsymbol{\Sigma}=E[(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^T]$分别为$\mathbf{x}$的均值和协方差矩阵。

二、高斯混合模型（GMM）

高斯混合模型[21]是用高斯概率密度函数精确地量化事物，将一个事物分解为若干的基于高斯概率密度函数（正态分布曲线）形成的模型。它可以看作是由 M 个单高斯模型组合而成的模型来描述复杂的数据分布（比如复杂的语音数据分布），如图 3-11，这 M 个子模型是混合模型的隐变量（Hidden variable）。一般来说，一个混合模型可以使用任何概率分布，这里使用高斯混合模型是因为高斯分布具备很好的数学性质以及良好的计算性能。

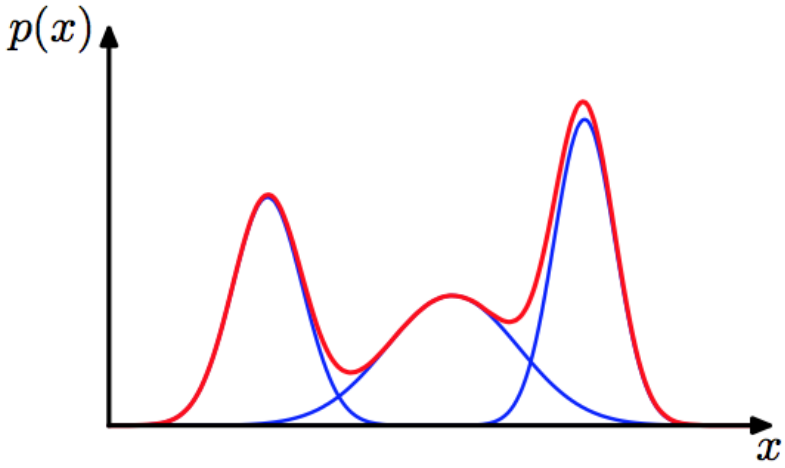


图 3-11 高斯混合模型

对于 M 个高斯模型的混合模型，c_m是观测数据属于第 m 个子模型的先验概率（混合权重），则高斯混合模型的概率分布为：

$$P(\mathbf{x}|\theta) = \sum_{m=1}^{M} c_m N(\mathbf{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{3.39}$$

$$N(\mathbf{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) = \frac{1}{(2\pi)^{\frac{D}{2}}(\boldsymbol{\Sigma}_m)^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_m)^T \boldsymbol{\Sigma}_m^{-1}(\mathbf{x}-\boldsymbol{\mu}_m)\right\} \tag{3.40}$$

其中，$\sum_{m=1}^{M} c_m = 1$，$N(\mathbf{x}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$是第 m 个子模型的高斯分布密度函数。

在语音处理中，我们通常用连续的概率密度函数来描述连续的观测向量（$\mathbf{o}_t \in R^D$）的概率分布。混合高斯模型是其中应用最成功的、最广泛的概率密度函数，若用混合高斯模型（GMM）来表示$\mathbf{o}_t$的概率分布

$$B_i(\mathbf{o}_t) = \sum_{m=1}^{M} \frac{c_{I,m}}{(2\pi)^{\frac{D}{2}}|\boldsymbol{\Sigma}_{I,m}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_{I,m})^T \boldsymbol{\Sigma}_{I,m}^{-1}(\mathbf{o}_t - \boldsymbol{\mu}_{I,m})\right\} \tag{3.41}$$

其中，$c_{I,m}$是状态 i 时第 m 个高斯子模型的权重，$\boldsymbol{\mu}_{I,m}$和$\boldsymbol{\Sigma}_{I,m}$为状态 i 时第 m 个高斯分布的均值向量和协方差矩阵。

三、基于 GMM-HMM 模型的语音识别技术

在语音识别中，第一个被广泛使用的统计模型是基于混合高斯模型的隐马尔可夫模型，即 GMM-HMM[15][22]。GMM-HMM 是一个统计模型，它描述了两个互相依赖的随机过程，一个是可观察的过程，另一个是隐藏的马尔可夫过程。一个 GMM-HMM 模型的参数集合由一个状态的先验概率向量、一个状态转移概率矩阵和一个状态相关的混合高斯模型参数组成。在语音建模中，GMM-HMM 中的一个状态通常与语音中的音素子段相关联。而每个状态都有一个概率密度分布，用 GMM 来描述。

基于单音素的 GMM-HMM 语音识别系统中 HMM 的一个状态对应一个音素。每个音素使用经典的三状态结构（图 3-12）。

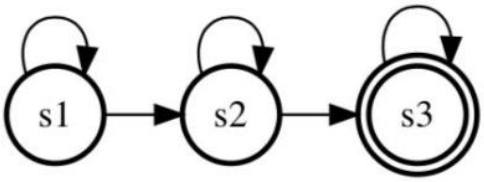


图 3-12 单音素的三状态结构

然而，单音素建模没有考虑协同发音效应。协同发音(coarticulation)是二十世纪 60 年代后逐渐发现的不同于传统语音学的新现象，指在语流中，音段并非是静止的、分离的声音，音段会对相邻的音段产生影响。协同发音的一个解决方法是用三音素建模，考虑音素的上下文，表示为“l-c+r”，其中，c 表示中心单元，l 为左相关信息，r 为右相关信息。假设共有 N 个音素，那么就需要N^3个三音素，当 N 增大时，参数量也是非常大的。因此，在语音建模中，使用决策树来进行三音素建模。这一过程依赖于单音素建模后的对齐。

决策树在机器学习中是一个预测模型；他代表的是对象属性与对象值之间的一种映射关系。树中每个结点表示某个对象，而每个分叉路径则代表某个可能的属性值，而每个叶结点则对应从根结点到该叶结点所经历的路径所表示的对象的值。决策树仅有单一输出，若欲有复数输出，可以建立独立的决策树以处理不同输出。语音建模中所使用的决策树是一个二叉树，每个非叶子结点上都会有个问题，叶子结点是绑定一个三音素的集合，绑定的粒度为状态，如将 A-B+C 和 A-B+D 的第一个状态绑在一起，第二、三个状态不一定绑在一起。图 3-13 是三音素决策树的一个例子，描述音素“-zh+”的三音素集合。如图所示，首先判断音素左边是否为元音音素，若是，则搜索决策树左边分支，若不是，则搜索决策树右边分支，以此类推，直至搜索至叶子结点。

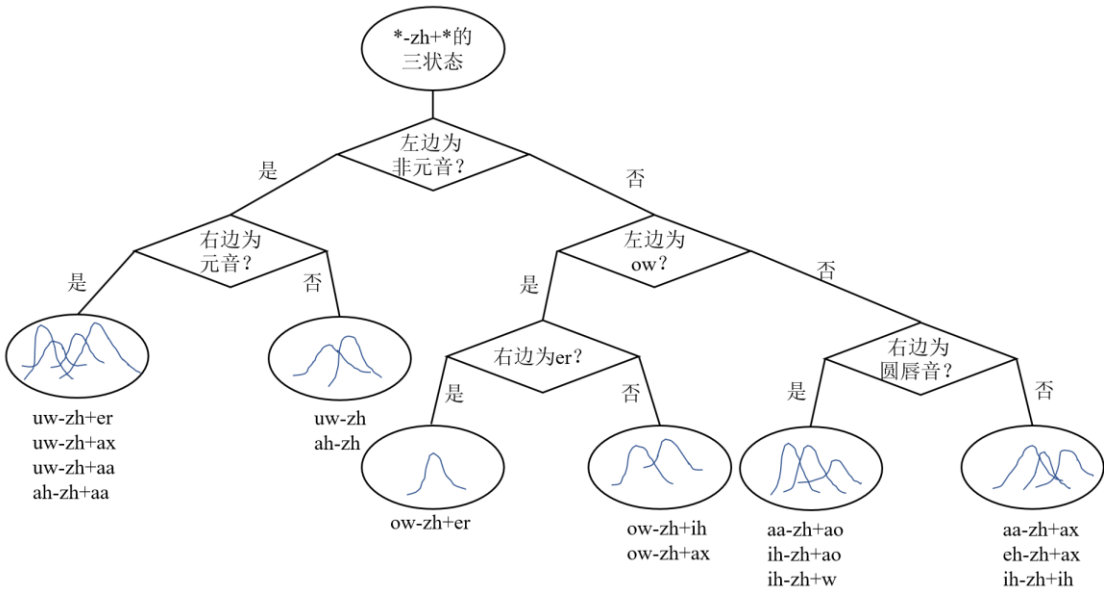


图 3-13 决策树示意图（引自[23]）

决策树的分裂依赖于问题级的设计，为了定义问题级，应先来确认划分特征，它包含两大类，发音相似性和建模单元的上下文关系信息[16]。问题集的构建通常由语言学家完成。比较常见的问题集是将音素划分为：爆破音（Stop）、鼻音（Nasal）、摩擦音（Fricative）、流音（Liquid）、元音（Vowel）等。

建立问题集后，就可以构建决策树。考虑到在单音素的经典三状态结构中，第一个状态和最后一个状态分别为起始状态和结束状态，它们只是在模型中起辅助作用，因此真正起作用的是中间的几个状态。在构造决策树时，一般只考虑中间的几个状态。

决策树是由自顶向下的顺序生成。构建决策树一般需要以下三步：

1）首先将所有的状态放入根结点中，然后进行结点分裂。结点分裂依赖于评估函数。决策树的评估函数用来估计决策树的结点上的样本相似性。可以选择对数似然概率作为评估函数

2）在每个结点进行分裂时，可以从问题集中选择一个问题，然后根据此问题把结点分成两个子结点，并且计算评估函数的增量。可以选择具有最大增量的问题，然后根据此问题把结点划分为两部分。

3）重复第二步，直至所有问题的增量都低于某个阈值的时候，结点上的分裂过程停止。最终，同一个叶子结点中的状态将被共享捆绑到一起。

可以看出，阈值大小会影响最终共享的结果。阈值越大，最终每个叶子结点中的状态越多，共享的程度就越高。这样最终模型的大小也就越小，但某些发音越有可能出现混淆。

由以上介绍可以看出，决策树的生成依赖于单音素建模后的对齐。因此，在进行三音素建模训练时需要先进行单音素训练。

对于语音识别系统，希望最大化$\mathrm{P}(W|\boldsymbol{X})=\frac{\mathrm{P}(\boldsymbol{X}|W)\mathrm{P}(W)}{\mathrm{P}(\boldsymbol{X})}$这个概率，其中$\boldsymbol{X}$是输入的语音特征，$W$则是识别出的文本（单词序列），$\mathrm{P}(W)$是语言模型，$\mathrm{P}(\boldsymbol{X}|W)$就是 GMM-HMM 要进行建模的声学模型。GMM-HMM 是生成模型，无法直接计算$\mathrm{P}(W|\boldsymbol{X})$并识别最佳单词序列。通常，利用贝叶斯法则对$\mathrm{P}(W|\boldsymbol{X})$进行转化：

$$\widehat{W} = \underset{W}{\operatorname{argmax}}\, \mathrm{P}(W|\boldsymbol{X}) = \underset{W}{\operatorname{argmax}} \frac{\mathrm{P}(\boldsymbol{X}|W)P(W)}{\mathrm{P}(\boldsymbol{X})} = \underset{W}{\operatorname{argmax}}\, \mathrm{P}(\mathbf{X}|W)P(W) \tag{3.42}$$

其中，$\mathrm{P}(\boldsymbol{X})$为先验概率，其数值不受声学模型和语言模型的影响，在计算最大值的过程中可以省略。$\mathrm{P}(\boldsymbol{X}|\mathrm{W})$可以通过本小节介绍的 GMM-HMM 或者 3.2.4 小节介绍的 DNN-HMM 计算，$\mathrm{P}(W)$通过 3.3 节的语言模型进行计算，求解声学模型和语言模型联合概率达到最大的最佳单词序列的过程通过 3.4 节的解码算法来实现。

3.2.3 深度神经网络简介

从 2010 年前后开始，深度神经网络技术便在多种语音领域展开应用并取得了优异的效果。深度神经网络技术的一个显著优势是可以通过网络的结构来建模随机高维向量之间的内在关系。经典的做法是先基于深度信念网络（Deep Belief Networks，DBN）做预训练，之后用反向传播（Backpropagation，BP）算法微调，通常这种网络被称为深度神经网络（Deep Neural Network，DNN）。在此，我们将通过对 DNN 这一简单的神经网络的介绍来阐述神经网络的基本工作原理。

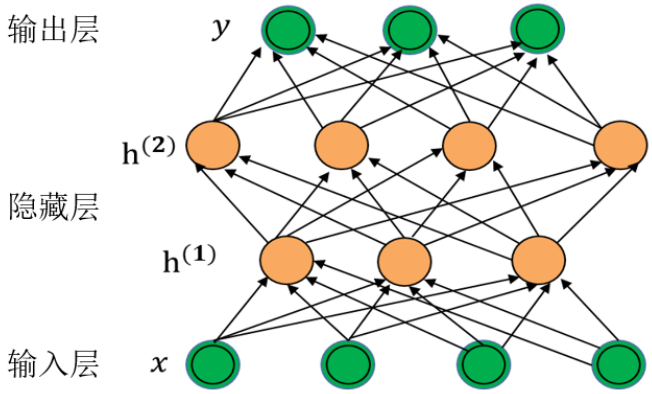


图 3-14 DNN 的基本结构

一个深度神经网络（Deep Neural Network，DNN）可以简单的定义为在输入层和输出层之间有多层隐含层（超过一层）的人工神经网络[24]。图 3-14 展示了一个具有两层隐含层的神经网络的基本结构。在每一个隐藏层，每一个神经元将会对来自上一层的所有神经元的输出在经过乘法的线性变换之后进行求和，最后再经过一个非线性的激活函数之后将所得的值当作此神经元传递给下一层。假如，选取 sigmod 函数$\mathrm{g}(\cdot)$作为激活函数，那么神经元的输出可以用如下公式表示:

$$\mathrm{h}_{\mathrm{j}}^{(\mathrm{l})} = \mathrm{g}\left(\mathrm{b}_{\mathrm{j}}^{(\mathrm{l})} + \sum_{\mathrm{i}} \mathrm{h}_{\mathrm{i}}^{(\mathrm{l}-1)} \mathrm{w}_{\mathrm{ij}}^{(\mathrm{l})}\right) \tag{3.43}$$

其中，$\mathrm{h}_{\mathrm{j}}^{(\mathrm{l})}$是第l个隐含层的第 j 个神经元的激活值，$\mathrm{h}_{\mathrm{i}}^{(0)} = \mathrm{x}_{\mathrm{i}}$表示的输入特征的第 i 维，

$b_j^{(l)}$则表示第l层的第j个神经元的偏置。同时，$w_{ij}^{(l)}$表示的是$h_i^{(l-1)}$到$h_j^{(l)}$之间的权重。激活函数的选取则依赖所要解决的具体问题。针对分类任务，输出层通常采用 softmax 函数，如下公式所示：

$$y_j = \frac{\exp\left(b_j^{(L+1)} + \sum_i h_i^{(L)} w_{ij}^{(L+1)}\right)}{\sum_k \exp\left(b_k^{(L+1)} + \sum_i h_i^{(L)} w_{ik}^{(L+1)}\right)} \tag{3.44}$$

其中，$\tilde{y}_j = h_j^{(L+1)}$可以理解为属于第j类的后验概率，而 L 则表示隐层的个数。对于回归任务，输出层则会考虑采用线性函数，如下公式所示：

$$\tilde{y}_j = b_j^{(L+1)} + \sum_i h_i^{(L)} w_{ij}^{(L+1)} \tag{3.45}$$

一个具有 L 个隐层的神经网络包含如下参数$\lambda = \{b^{(1)}, W^{(1)}, \ldots, b^{(L+1)}, W^{(L+1)}\}$。这些参数可以通过最小化真实值和预测值之间的误差来进行优化，优化的过程需要用到误差反向传播算法[25]。例如，对于分类任务来说，真实分类和输出预测分类的后验概率的交叉熵一般作为损失函数，如下公式所示：

$$L(W, b, y, \tilde{y}) = -\sum_j y_j \log(\tilde{y}_j) \tag{3.46}$$

其中，y_j表示输入所对应类别为j的真实概率。对于分类任务来说，概率值通常是 0 或 1，1 表示属于这一类，0 表示不属于这一类。对于回归任务，误差函数可以用式（3. 47）的均方误差来表示,

$$L(W, b, y, \tilde{y}) = \sum_j (\tilde{y}_j - y_j)^2 \tag{3.47}$$

其中，y_j和$\tilde{y}_j$分别表示第 j 维的预测值和真实值。此外，对于回归任务来说，DNN 也可以认为是一种给定x下关于y的条件概率模型，如公式（3. 48）所示：

$$P(y|x, \lambda) = N(y; \tilde{y}, I) \tag{3.48}$$

其中，I 是一个单位矩阵，λ表示神经网络模型的参数，$\tilde{y}$依赖于输入x和参数λ。因此最小化y和$\tilde{y}$之间均方根误差的过程，类似于对参数λ的极大似然估计。

DNN 是输入和输出之间高度复杂和非线性关系的强大模型。训练带有多个隐藏层的

DNN需要前向传播和反向传播。DNN的前向传播算法比较简单。它就是利用若干个权重系数矩阵$W^{(l)}$和偏置向量$b^{(l)}$以及输入值向量$x(=h^{(0)})$进行一系列线性运算和激活运算，从输入层开始，一层层地向前计算，一直到运算到输出层，得到输出结果为止。即，对一个具有 L 个隐层的神经网络，l从 1 到 L 层循环执行式(3. 44) 的运算，在 L+1 的输出层根据问题类型执行式(3. 45)和(3. 46)的运算。

为了优化 DNN 模型，我们必须把输出层的误差反向传递到最底层。为了方便且不失一般性，在此我们将模型的均方差损失函数改写为下式，

$$\begin{aligned} L(\mathbf{W},\mathbf{b},\mathbf{x},\mathbf{y}) &= \frac{1}{2}\left\|g\left(\mathbf{W}^{(L+1)}\mathbf{h}^{(L)}+\mathbf{b}^{(L+1)}\right)-\mathbf{y}\right\|_2^2 \\ &= \frac{1}{2}\left\|g(\mathbf{Z}^{L+1})-\mathbf{y}\right\|_2^2 \\ \text{其中，}\ \mathbf{Z}^{L+1} &= \mathbf{W}^{(L+1)}\mathbf{h}^{(L)}+\mathbf{b}^{(L+1)} \end{aligned} \tag{3.49}$$

通过对损失函数L分别求$\mathbf{W}$和$\mathbf{b}$的偏微分，我们可以得到$\mathbf{W}$和$\mathbf{b}$的梯度，

$$\begin{aligned} \frac{\partial L(W,b,x,y)}{\partial W^{L+1}} &= \frac{\partial L(W,b,x,y)}{\partial Z^{L+1}}\frac{\partial Z^{L+1}}{\partial W^{L+1}} = \frac{\partial L(W,b,x,y)}{\partial h^{L+1}}\frac{\partial h^{L+1}}{\partial z^{L+1}}\frac{\partial Z^{L+1}}{\partial W^{L+1}} \\ &= (h^{L+1}-y)\odot g'(z^{L+1})(h^{L})^{T} \end{aligned} \tag{3.50}$$

$$\begin{aligned} \frac{\partial L(W,b,x,y)}{\partial b^{L+1}} &= \frac{\partial L(W,b,x,y)}{\partial Z^{L+1}}\frac{\partial Z^{L+1}}{\partial b^{L+1}} = \frac{\partial L(W,b,x,y)}{\partial h^{L+1}}\frac{\partial h^{L+1}}{\partial z^{L+1}}\frac{\partial Z^{L+1}}{\partial b^{L+1}} \\ &= (h^{L+1}-y)\odot g'(z^{L+1}) \end{aligned} \tag{3.51}$$

其中，$\odot$代表 Hadamard 积，对于两个维度相同的向量$A=[a_1,a_2,\dots,a_n]^T$和$B=[b_1,b,\dots,b_n]^T$，则$A\odot B=[a_1b_1,a_2b_2,\dots,a_nb_n]^T$。

我们注意到在求解输出层的 $\boldsymbol{W}$,$\boldsymbol{b}$ 的时候，有一个公共的部分$\frac{\partial L(W,b,x,y)}{\partial Z^{L+1}}$，因此可以把公共的部分即对$Z^{L+1}$ 先算出来，记为：

$$\delta^{L+1}=\frac{\partial L(W,b,x,y)}{\partial Z^{L+1}}=(h^{L+1}-y)\odot g'(z^{L+1}) \tag{3.52}$$

则

$$\begin{aligned} \delta^{l}=\frac{\partial L(W,b,x,y)}{\partial Z^{l}} &= \frac{\partial L(W,b,x,y)}{\partial Z^{L+1}}\frac{\partial z^{L+1}}{\partial z^{L}}\frac{\partial z^{L+1}}{\partial z^{L-1}}\cdots\frac{\partial z^{l+1}}{\partial z^{l}} \\ &= \frac{\partial L(W,b,x,y)}{\partial Z^{l+1}}\frac{\partial z^{l+1}}{\partial z^{l}} \qquad ,\ \{l=L,L-1,\dots,1\} \\ &= \delta^{l+1}\frac{\partial z^{l+1}}{\partial z^{l}} \end{aligned} \tag{3.53}$$

根据上式，我们可以很方便的计算出第l层的W^l 和b^l的梯度，

$$g_{W^l} = \frac{\partial L(W,b,x,y)}{\partial W^l} = \frac{\partial L(W,b,x,y)}{\partial Z^l}\frac{\partial Z^l}{\partial W^l} = \delta^l(h^{l-1})^T \tag{3.54}$$

$$g_{b^l} = \frac{\partial L(W,b,x,y)}{\partial b^l} = \frac{\partial L(W,b,x,y)}{\partial Z^l}\frac{\partial Z^l}{\partial b^l} = \delta^l \tag{3.55}$$

那么，在时间t梯度下降的更新可以用(3. 56)，(3. 57)来表示，

$$W_t^l = W_{t-1}^l - \eta_k g_{W^l} \tag{3.56}$$

$$b_t^l = b_{t-1}^l - \eta_k g_{b^l} \tag{3.57}$$

其中，η_k被称为是步长或者是学习率, $l = \{L, L-1, \dots, 1\}$。

3.2.4 基于深度学习的声学模型

在 20 世纪 80 年代末，人们开始使用 HMM [26][27] 的方法来进行语音识别，这个方法可以解决 GMM 不能直接为时序连续的语音信号建模的问题 [28][29]。基于 GMM-HMM 的混合模型在大词汇连续语音识别系统中被认为是一种非常有前景的技术。 随着人工神经网络（ANN）[26]的兴起，人们开始利用上下文无关的音素状态作为 ANN[26] 训练的标注信息，但只适用于小词汇任务。后来，逐渐扩展到上下文相关的音素建模，并来时用于中大型词表的自动语音识别任务。一开始的人工神经网络多为浅层网络，随着深度学习的进一步发展，深度神经网络（DNN）逐渐用于声学模型的搭建，因而基于 DNN 的混合模型逐渐代替 GMM-HMM 成为主流声学模型。

在基于深度学习的语音识别中，最先出现的是混合声学模型，例如，基于深度神经网络的 DNN-HMM 声学模型，基于循环神经网络的 RNN-HMM 声学模型，以及基于卷积神经网络的 CNN-HMM 声学模型等等。以上提及的声学模型均为深度学习在传统语音识别领域的应用。在 21 世纪 10 年代，基于深度学习的端到端模型逐渐崭露头角，其一开始在机器翻译领域广泛应用，随后扩展到语音识别领域开创了端到端语音识别的新时代。

3.2.4.1 基于深度神经网络（DNN）模型[21][30]

在 20 世纪 80 年代末，人们开始使用人工神经网络 [21]以及 HMM[26] 的方法来进行语音识别，这个方法可以解决 GMM 不能直接为时序连续的语音信号建模的问题[28][29]。

神经网络是基于感知机的扩展，而 DNN 可以理解为有很多隐藏层的神经网络。从 DNN 按不同层的位置划分，DNN 内部的神经网络层可以分为三类，输入层，隐藏层和输出

层, 如图 3-14 示例。基于 DNN-HMM 的混合语音识别系统[21][30]的结构如下图 3-15 所示。其中，HMM 可以用于描述语音信号的动态变化，来解决语音信号长度变化的问题。DNN 则估计观测特征的概率。首先给定声学的观测特征，然后用 DNN 的输出节点来估计 HMM 某个状态的后验概率。在解码过程中，则可以运用高效的维特比算法。

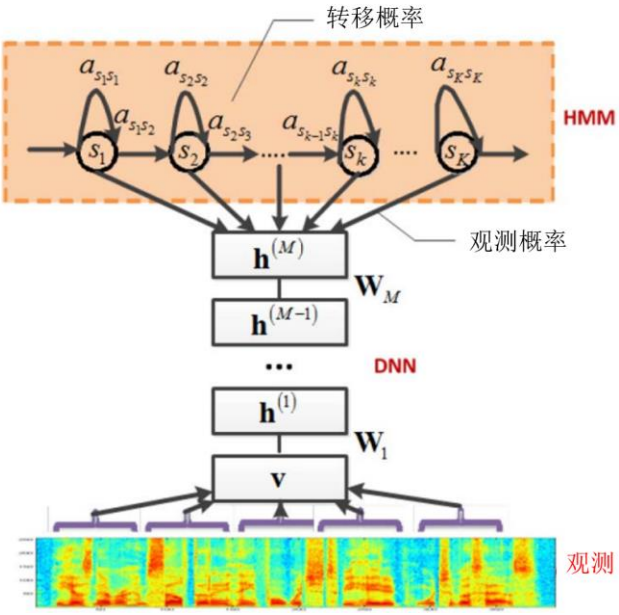


图 3-15 DNN-HMM 框架图（引自[21]）

在语音识别系统中，通过最大化P($\boldsymbol{X}|W$)P(W)求解最佳单词序列。其中 $\boldsymbol{X}$ 是输入的语音特征，W 则是识别出的文本（单词序列），P(W)是语言模型，P($\boldsymbol{X}|W$)是声学模型。一般情况下，语言模型P(W)的模型训练和概率计算是独立于声学模型的。每个单词由子词声学模型并且每个子词模型由多个状态组成。因此，单词序列 W 可以由状态序列 $\boldsymbol{Q}=\boldsymbol{q_1 q_2},\dots,\boldsymbol{q_T}$替换，即P($\boldsymbol{X}|W$)= P($\boldsymbol{X}|\boldsymbol{Q}$)。在基于 GMM-HMM 的语音识别系统，P($\mathbf{X}|\mathbf{Q}$)声学模型的概率可以用如下的状态转移概率和观测值概率（每个状态上的$P(\mathbf{x}_t|q_t)$通过 GMM 计算）来求解近似值（最佳状态所对应的概率值）：

$$P(X|\boldsymbol{Q}) \approx \max_{q_1\cdots q_T} \prod_{t=1}^{T} P(q_t|q_{t-1})\, P(\mathbf{x}_t|q_t) \quad (3.58)$$

其中，$\mathbf{x}_t$是第 t 帧的特征向量。在 HMM 的解码过程中，系统需要似然度$P(\mathbf{x}_t|q_t)$，而不是后验概率。在基于 DNN-HMM 的语音识别系统中，DNN 能直接计算后验概率$P(q_t|\mathbf{x}_t)$而无法直接计算似然度$P(\mathbf{x}_t|q_t)$。因此，我们需要把后验概率转化为似然度：

$$P(\mathbf{x}_t|q_t) = \frac{P(q_t|\mathbf{x}_t)P(\mathbf{x}_t)}{P(q_t)} \quad (3.59)$$

其中，$P(\mathbf{x}_t)$的数值不受声学模型和语言模型的影响，在求解最大值过程中可以省略。$P(q_t)$是从训练集中统计的每个状态的先验概率。在 DNN-HMM，$P(q_t|\mathbf{x}_t)$由 DNN 的 softmax 函数直接计算得到。在 DNN-HMM 中，对于所有的状态，我们只训练一个完整的 DNN 来估计状态的后验概率$P(q_t|\mathbf{x}_t)$。而在传统的 GMM-HMM 中，对于不同的状态我们用

不同的 GMM 建模。通过以上的处理，DNN-HMM 和 GMM-HMM 可以采用相同的解码算法。

在 DNN 的训练过程中，标签是必不可少的，即要有输入帧与状态的对应。通常利用训练好的 GMM-HMM 模型，进行维特比解码后就可得到使得当前句子生成概率最大的状态序列，每一帧对应一个状态标注。这些成对的（$\boldsymbol{X}$,$\boldsymbol{Q}$）可以被用于 DNN 模型的训练，即输入“当前帧”（可以包含左右的上下文关系），输出则是 HMM 中N个状态的后验概率，通常利用交叉熵（CE）损失函数来进行优化。接着利用训练好的 DNN 重新估计 HMM 的转移概率参数，同时对状态进行二次标注，再对 DNN 进行训练。如此往复，直至收敛。

通常，上下文相关的模型在一些任务中会优于 GMM-HMM，但改善并不显著。为了大幅度的提升识别性能，可以做出以下几个改变：首先，将浅层神经网络替换为深层神经网络。其次，可以使用绑定后的三音素状态代替单音素状态作为神经网络的输出单元。这种模型被称为 CD-DNN-HMM 模型。下面为大家介绍几种常用的深度神经网络模型。

1） 基于前馈深度神经网络（FF-DNN）的声学模型[31]

前馈深度神经网络(Feed Forward Deep Neural Network, FF-DNN)，是典型的深度学习模型。在 FF-DNN 内部，参数从输入层向输出层单向传播，有异于递归神经网络，它的内部不会构成有向环。 在前馈网络中，信息总是朝着一个方向移动。因此，它不同于其后代：递归神经网络。前馈神经网络是设计的第一种也是最简单的人工神经网络。在该网络中，信息仅在一个方向上移动，即从输入节点向前经过隐藏节点（如果有）并到达输出节点，网络中没有循环。在 FF-DNN 模型中，输入是除了当前帧之外，还包括相邻的几帧的信息，通过连接操作将这几帧合并为新的当前帧。

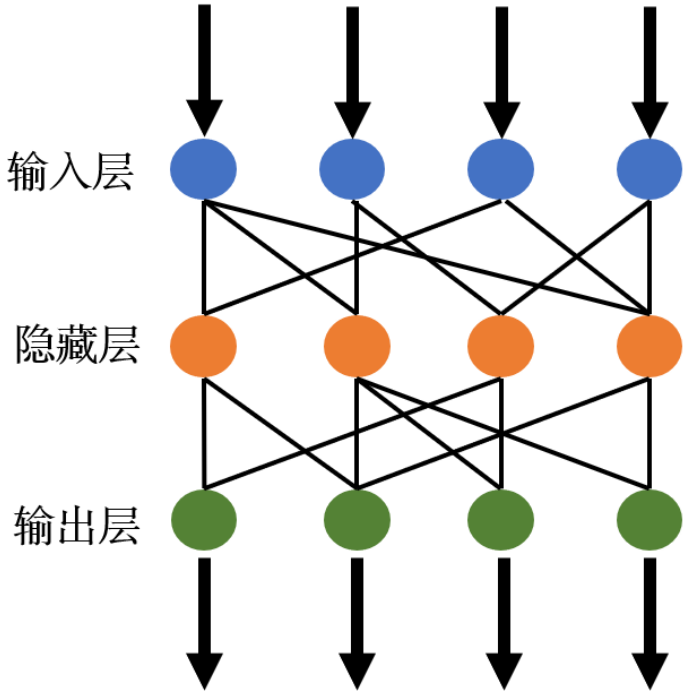


图 3-16 前馈深度神经网络

从图中可以看出，在 FF-DNN 模型中，输入是除了当前帧之外，还包括相邻的几帧的信息，通过连接操作将这几帧合并为新的当前帧。

2） 基于上下文相关神经网络（CD-DNN）的声学模型[32][33]

早期的 ANN-HMM，通常只使用上下文无关的音素（音素是语音的最小单位）状态作为

ANN 训练的标注信息，随着浅层的神经网络被替换成 DNN，其次使用聚类后的状态（绑定后的三音素状态）代替单音素状态作为神经网络的输出单元，这种改善后的 ANN-HMM 混合模型称为 CD-DNN-HMM。因此，基于上下文相关的深度学习声学模型在前馈神经网络的基础上进一步利用了上下文相关的信息进行训练。

在 CD-DNN-HMM 中，对于所有的状态，我们只训练一个完整的 DNN 来估计状态的后验概率，这与传统的 GMM 是不同的，因为 GMM 框架下，我们会使用多个不同的 GMM 对不同的状态建模。除此之外，典型的 DNN 输入不是单一的一帧，而是一个 2w+1 帧大小的窗口特征，这使得相邻帧的信息可以被有效的利用。

由 CD-DNN-HMM 最终解码出的字词序列需要同时考虑到声学模型和语言模型的概率，通过权重系数 λ 去平衡二者之间的关系。训练 CD-DNN-HMM 的第一步通常就是使用无监督的训练数据训练一个 GMM-HMM 系统，前文提到 DNN 训练标注是由 GMM-HMM 系统采用维特比算法产生得到的，因此标注的质量会影响 DNN 系统的性能。一旦训练好 GMM-HMM 模型 hmm0，就可以创建一个从状态名到三音素的映射。然后利用 hmm0 采用维特比算法生成一个状态层面的强制对齐，以生成从特征到三音素的映射对，为 DNN 提供标注好的训练数据。

3） 基于时延神经网络（TDNN）的声学模型[34]

作为卷积神经网络的前身，时延神经网络（Time-Delay Neural Network，TDNN）是为了解决语音识别中传统方法 HMM 无法适应语音信号中的动态时域变化。TDNN 的两个明显的特征是动态适应时域特征变化和参数较少，区别于传统的深度神经网络中输入层与隐含层一一连接的特征，TDNN 中隐含层的特征不仅与当前时刻的输入有关，而且还与未来时刻的输入有关，其目的是：1）使用不变性对模式进行分类，以及 2）在网络的每一层建模上下文。

不变移位分类意味着分类器在分类之前不需要显式分割。 对于时间模式（例如语音）的分类，TDNN 因此避免了在对声音进行分类之前必须确定声音的起点和终点。对于 TDNN 中的上下文建模，每一层的每个神经单元不仅从下一层接收输入，而且从单元输出及其上下文接收输入。如下图所示：

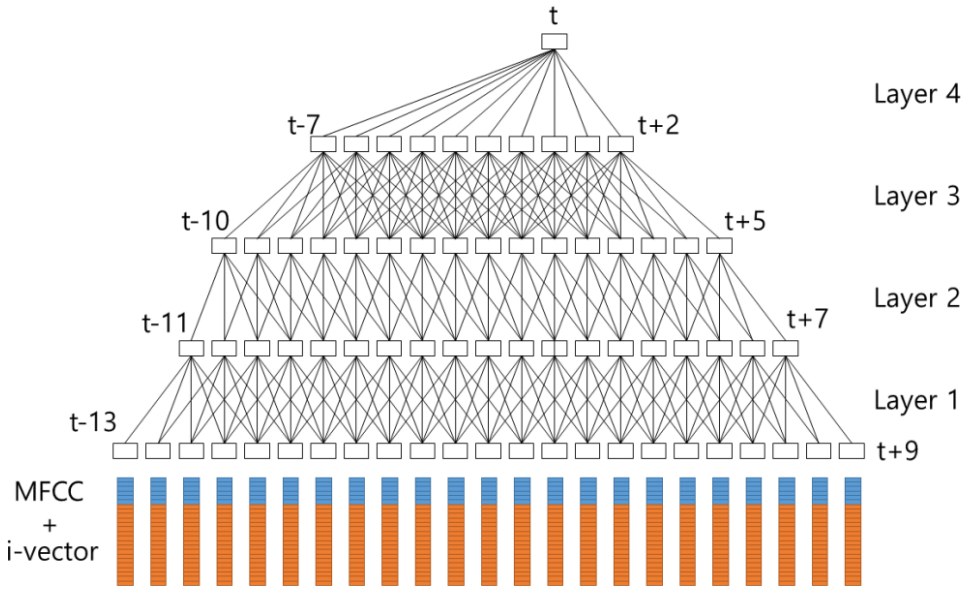


图 3-17 时延神经网络结构（引自[35]）

与常规多层感知器不同，TDNN 中每一层的所有单元都从下一层输出的上下文窗口中获取输入。对于语音信号，每个单元都与下面单元的输出相连，但也与这些相同单元的延时（过去）输出相连。

TDNN 在大概 1987 年用于解决语音识别问题，最初专注于不变式音素识别。 语音非常适合 TDNN，因为语音的长度很少一致，很难或不可能进行精确的分段。 通过扫描过去和将来的声音，TDNN 能够以时移不变的方式为该声音的关键元素构建模型。这尤其有用，因为声音会通过混响抹去。大型语音 TDNN 可以通过预训练和组合较小的网络来模块化构建。

3.2.4.2 基于循环神经网络模型(RNN) [36][37][38]：

神经网络在语音识别方面具有悠久的历史，通常与隐马尔可夫模型结合使用。深层前馈网络在声学建模方面的巨大进步，引起了人们的关注。鉴于语音是一个动态过程，因此考虑将递归神经网络（RNN）作为声学模型。

给定输入序列$\boldsymbol{x} = (x_1, \ldots, x_T)$，标准循环神经网络 (RNN) 计算隐藏向量序列 $\boldsymbol{h} = (h_1, \ldots, h_T)$ 和输出向量序列 $\boldsymbol{y} = (y_1, \ldots, y_T)$ 从 t=1 到 T 迭代以下等式：

$$\mathrm{h_t} = \mathrm{H}(\mathrm{W_{xh}x_t} + \mathrm{W_{hh}h_{t-1}} + \mathrm{b_h}) \tag{3.60}$$

$$\mathrm{y_t} = \mathrm{W_{hy}h_t} + \mathrm{b_y} \tag{3.61}$$

其中 $\boldsymbol{W}$ 项表示权重矩阵（例如 $\mathrm{W_{xh}}$是输入隐藏权重矩阵），b项表示偏置向量（例如b_h是隐藏偏置向量），H是隐藏层函数。

在实际的语音识别任务中，深度循环神经网络效果通常更加优异。通过增加隐藏层的个数，可以得到深度神经网络。具体的实现方式有一下几种：1）在循环层之前加入多个普通的前馈层；2） 增加至多个循环层，并且结合当前隐藏层上一时刻的结果来计算当前时刻隐含层的输出；3）在循环层与输出层之间增加多个前馈层，使得输入向量先进行多层映射。

通过训练得到循环神经网络的参数，RNN 的训练样本都是时间序列。通常使用延时反向传播算法（BPTT）来进行 RNN 模型训练。与全连接网络在层之间递推的方式不同的是，BPTT 算法是沿着时间轴来进行递推。然而，在梯度反向传播时，循环神经网络面临着严重的梯度消失与梯度爆炸问题。当输入序列过长时，很难对梯度进行有效更新。为了解决这个问题，人们提出了一个新的模型。

1） 长短时记忆模型（LSTM）[38][39][40]

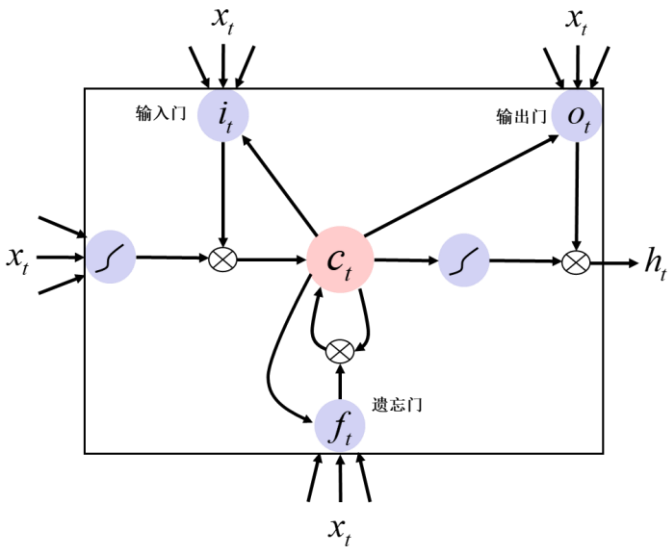


图 3-18 长短时记忆模型结构（引自[38]）

基于 RNN 网络，Schmidhuber 等人在 1997 年提出了长短时记忆模型（LSTM）[7]。通过使用输入门、输出门这两个门来计算$\mathbf{h}$。后来研究者加入了遗忘门来更好的计算$\mathbf{h}$。因此现在的 LSTM 网络通常包含三个门。

公式（3.60）中的 **H** 通常是 sigmoid 函数。 然而，长短期记忆 (LSTM) 架构使用专门构建的记忆单元来存储信息，更擅长发现和利用远程上下文。 图 3-18 表示单个 LSTM 存储单元。**H** 则由以下复合函数实现：

$$\mathrm{I_t} = \sigma(\mathrm{W_{xi}x_t} + \mathrm{W_{hi}h_{t-1}} + \mathrm{W_{ci}c_i} + \mathrm{b_i}) \tag{3.62}$$

$$\mathrm{F_t} = \sigma(\mathrm{W_{xf}x_t} + \mathrm{W_{hf}h_{t-1}} + \mathrm{W_{cf}c_{t-1}} + \mathrm{b_f}) \tag{3.63}$$

$$\mathrm{c_t} = \mathrm{f_t c_{t-1}} + \mathrm{i_t}\tanh(\mathrm{W_{xc}x_t} + \mathrm{W_{hc}h_{t-1}} + \mathrm{b_c}) \tag{3.64}$$

$$\mathrm{o_t} = \sigma(\mathrm{W_{xo}x_t} + \mathrm{W_{ho}h_{t-1}} + \mathrm{W_{co}c_t} + \mathrm{b_o}) \tag{3.65}$$

$$\mathrm{h_t} = \mathrm{o_t}\tanh(\mathrm{c_t}) \tag{3.66}$$

其中σ是 logistic sigmoid 函数，i、f、o和c分别是输入门、遗忘门、输出门和记忆单元向量，它们的大小都与隐藏向量h相同。 从记忆单元到门向量（例如$\mathrm{W_{si}}$）的权重矩阵是对

角线的，因此每个门向量中的元素m仅接收来自记忆单元向量的元素m的输入。

2）双向循环模型（BRNN） [38][41][42]:

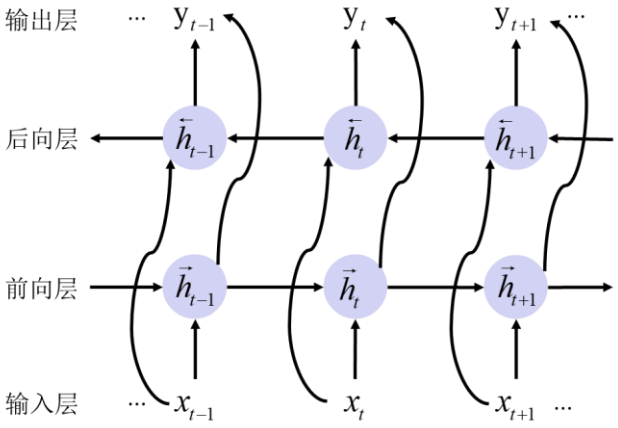


图 3-19 双向循环神经网络模型结构(引自[38])

传统 RNN 的一个缺点是它们只能利用先前的上下文。在语音识别中，一次转录整个话语，于是人们想到利用未来的上下文。 双向 RNN (Bidirectional RNNs，BRNN) 通过使用两个单独的隐藏层在两个方向上处理数据来实现这一点，然后将它们前馈传播到相同的输出层。 如图 3-19 所示，BRNN 通过从 t=T 到 1 迭代后向层，从 t=1 到 T 迭代前向层，然后计算前向隐藏序列$\vec{\mathbf{h}}$，后向隐藏序列$\overleftarrow{\mathbf{h}}$和输出序列$\mathbf{y}$更新输出层：

$$\mathrm{h_t} = \mathrm{H}(\mathrm{W_{hh}h_{t-1}} + \mathrm{b_h}) \tag{3.67}$$

$$\mathrm{h_t} = \mathrm{H}(\mathrm{W_{xh}x_t} + \mathrm{W_{hh}h_{t-1}} + \mathrm{b_h}) \tag{3.68}$$

$$\mathrm{y_t} = \mathrm{W_{hy}h_t} + \mathrm{W_{hy}h_t} + \mathrm{b_y} \tag{3.69}$$

将 BRNN 与 LSTM 相结合得到双向 LSTM，它可以在两个输入方向访问长序上下文。

混合 HMM 神经网络系统取得成功的一个关键因素是使用深度架构，它能够逐步建立更高级别的声学特征表示。 因此，可以通过将多个 RNN 隐藏层堆叠在一起来创建深度 RNN，一层的输出序列构成下一层的输入序列。

3.3 语言模型

从词表中任意选择若干同音词所构成的序列不一定能构成自然语言中的合乎句法的句子。人类在识别和理解语句时充分利用了这种约束。在语音识别中利用语言模型来实现这种约束。语言模型是根据语言客观事实而进行的语言抽象数学建模，用以计算文字序列的

概率，可以分为基于统计的 N-gram 语言模型和基于深度学习的神经语言模型。基于统计的语言模型是从大量的文本资料中统计出各个词的出现概率及其相互关联的条件概率，并将这些知识于声学模型匹配相结合进行结果判决，以减小由于声学模型不够合理而产生的错误识别。基于深度学习的语言模型是通过训练网络来学习词表征和单词序列的概率化表示。下面将详细介绍这两种语言模型。

在介绍语言模型前，我们需要先了解语言模型一些常见的术语[16]：

- 有声停顿(fillers/filled pauses)：停顿时发出的没有实际意义的音，如："I uh gave a re- report yesterday"中的"uh"。
- 截断(fragment)：表示说话没有说完整，如上面句子中的"re"。
- 词目(lemma)：词语主干(stem)相同，比如 dogs 和 dog 是一个词目。
- 词形(wordforms)：完整的词语样子，比如 dogs 和 dog 是两个词形。
- 型(type)：语料库或者字典中不同单词的数目。
- 例(token)：语料中单词数目。
- 字典(vocabulary)：语言模型的基本组件，规定了声学建模单元到语言建模单元的映射。

3.3.1 基于 N-gram 的语言模型

3.3.1.1 N-gram 语言模型[16]

设词序列向量为$W = w_1, w_2, \dots, w_Q$，则其概率可以表示为

$$\begin{aligned} P(W) &= P(w_1, w_2, \dots, w_Q) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1w_2) \dots P(w_Q|w_1w_2 \dots w_{Q-1}) \\ &= \prod_{i=1}^{Q} P(w_i|w_1w_2 \dots w_{i-1}) \\ &= \prod_{i=1}^{Q} P(w_i|w_1^{i-1}) \end{aligned} \tag{3.70}$$

然而，要可靠地估计出一种语言所有词在所有长度序列下的条件概率几乎是不可能的事，因为其计算的复杂度太高，并且一些稀疏词的概率难以计算。因此，出现了简化模型—N-gram 模型。

假设对式（3.71）的条件概率只考虑与前$N-1$个词相关，即为 N 元文法（N-gram）模型。

$$P(W) = \prod_{i=1}^{Q} P(w_i|w_{i-N+1} \dots w_{i-2}w_{i-1}) \tag{3.71}$$

在二元文法（bi-gram）中，假设词w_i出现的概率只与w_{i-1}有关，为了使$P(w_i|w_{i-1})$在$i = 1$时有意义，通常引入特殊标识<S>和</S>来表示整个句子的开始和结束。这样可以利用特殊标识将整个句子的概率调整为 1，在计算$P(w_i|w_{i-1})$时可以简化为计算$(w_{i-1}w_i)$在语料库中出现的次数占(w_{i-1})在语料库中出现次数的比例，即

$$\widehat{P}(w_{i-N+1} \dots w_{i-2}w_{i-1}) = \frac{c(w_{i-N+1}w_{i-N+2} \dots w_i)}{c(w_{i-N+1}w_{i-N+2} \dots w_{i-1})} \tag{3.72}$$

其中，c(W)表示词序列在训练数据中出现的次数。

困惑度（PPL）[Jelinek et al., 1977] 是一种用来衡量一个概率模型质量的信息论度量标准，是评价语言模型的一种方法。PPL 越低说明模型越好。给定一个包含 *N* 个单词的语料库和一个语言模型，该语言模型的 PPL 为：

$$\begin{aligned} PP(W) &= P(w_1, w_2, \dots, w_Q)^{-\frac{1}{Q}} \\ &= \sqrt[Q]{\frac{1}{P(w_1, w_2, \dots, w_Q)}} \\ &= \sqrt[Q]{\prod_{i=1}^{Q} \frac{1}{P(w_i | w_{1-N+1}^{i-1})}} \end{aligned} \tag{3.73}$$

3.3.1.2 平滑技术

由 N-gram 模型可知，即使在 *N* 较小的情况下，要统计的条件概率也是非常庞大的数字，因而常常会出现$c(W) = 0$或者接近零的情况，这样得到的结果将不可靠，解决这种训练数据稀疏的方法是采用一些平滑技术。其基本思想是将模型中可见事件的概率值进行折扣（discounting），并将该折扣值重新分布给不可见事件的元素序列，所以它可以保证模型中任何概率均不为零，且可以使模型参数概率分布趋向于更加均匀。因此，平滑方法由概率值折扣的策略和折扣值的分布方法所决定。

（a）加法平滑技术

这类方法是采取对所有（包括在模型出现或未出现的）事件的频率值加上一个固定的值来避免零概率事件。主要有两种方法。

一种是最简单的$add-\delta$平滑，即在 N-gram 模型中出现的每个事件的次数加上一个固定值

$$P_{add}(w_{i-N+1}^{i-1}) = \frac{c(w_{i-N+1} w_{i-N+2} \dots w_i) + \delta}{\sum_{w_i} c(w_{i-N+1} w_{i-N+2} \dots w_{i-1}) + \delta |V|} \tag{3.74}$$

其中，$0 \le \delta \le 1$，$|V|$表示模型中的元素个数。当$\delta = 1$时，该方法又称为“加 1 法”或“拉普拉斯平滑法”。这种平滑算法简单、易懂、易实现，但一般来说性能较差。

另一种是 one-count 平滑技术

$$P_{one}(w_{i-N+1}^{i-1}) = \frac{c(w_{i-N+1}^{i}) + \alpha P_{one}(w_i | w_{i-N+2}^{i-1})}{\sum_{w_i} c(w_{i-N+1}^{i-1}) + \alpha} \tag{3.75}$$

其中，α是常数。

（b）Good-Turing 估计

Good-Turing 估计对 N-gram 中出现 *r* 次的事件，假设它出现的次数为r^*从，即

$$r^* = (r+1)\frac{n_{r+1}}{n_r} \tag{3.76}$$

其中，n_r表示在 N-gram 训练集中出现 *r* 次的事件个数，则 N-gram 中出现 *r* 次的事件的条件概率为

$$P_{GT} = \frac{r^*}{N} \tag{3.77}$$

其中，*N* 为 N-gram 中所有 *N* 元对的总数，要注意的是

$$\begin{aligned} N &= \sum_{r=0}^{\infty} n_r\, r^* \\ &= \sum_{r=0}^{\infty} (r+1)\, n_{r+1} \\ &= \sum_{r=1}^{\infty} n_r r \end{aligned} \tag{3.78}$$

也就是说，N 等于这个分布中的最初的计数。这样，这个样本中所有事件的概率之和为

$$\sum_{r>0} n_r p_{GT} = 1 - \frac{n_1}{N} < 1 \tag{3.79}$$

由于 Good-Turing 估计不包括低阶模型对高阶模型的插值，通常不能单独作为一个 N-gram 平滑算法，而是作为其他平滑算法的一个计算工具。

（c）Katz 平滑技术

该平滑算法是当一个 N 元文法出现次数$c(w_{i-N+1}^{i})$足够大时，通过最大似然估计得到的$P_{ML}(w_{i-N+1}^{i})$是一个可靠的统计估计，而当$c(w_{i-N+1}^{i})$不够大是，采用 Good-Turing 估计对其进行进行折扣，并将折扣值赋给未出现的 *N* 元对，且补偿值与其低阶模型相关。当$c(w_{i-N+1}^{i}) = 0$时，按着低阶模型$P(w_{i-N+2}^{i})$比例来分配给未出现的 N 元对的概率，即一个词出现的次数更新为

$$C_{Katz}(w_{i-1}^{i}) = \begin{cases} d_r r, & r > 0 \\ \alpha w_{i-1} P_{ML}(w_I), & r = 0 \end{cases} \tag{3.80}$$

其中$d_r r$为词串出现 *r* 次时，平滑后的次数，d_r为不大于 1 的参数，则 *N* 元文法折扣后的次数为

$$C_{Katz}(w_{i-N+1}^{i}) = \begin{cases} d_r c(w_{i-N+1}^{I}), & c(w_{i-N+1}^{i}) > 0 \\ \alpha w_{i-N+2}^{i} c_{Katz}(w_{i-N+2}^{I}), & c(w_{i-N+1}^{i}) = 0 \end{cases} \quad (3.81)$$

经过 Katz 平滑后的概率值为

$$P_{Katz}(w_i|w_{i-N+1}^{i-1}) = \begin{cases} d_r P(w_i|w_{i-N+1}^{i-1}), & c(w_{i-N+1}^{i}) > 0 \\ \alpha(w_{i-N+1}^{i-1}) P_{Katz}(w_i|w_{i-N+2}^{i-1}), & c(w_{i-N+1}^{i}) = 0 \end{cases} \quad (3.82)$$

其中，$\alpha(w_{i-N+1}^{i-1})$的取值应该使事件分布的总数$\sum_{w_i} c_{Katz}(w_{i-N+1}^{i})$保持不变，即

$$\alpha(w_{i-N+1}^{i-1}) = \sum_{w_i} c_{Katz}(w_{i-N+1}^{i}) \quad (3.83)$$

其值为

$$\alpha(w_{i-N+1}^{i-1}) = \frac{1-\sum_{w_i: c(w_{i-N+1}^{i})>0} P_{Katz}(w_i|w_{i-N+1}^{i-1})}{1-\sum_{w_i: c(w_{i-N+1}^{i})>0} P_{Katz}(w_i|w_{i-N+2}^{i-1})} \quad (3.84)$$

在d_r的计算中，数目大的次数被认为是可靠，因此不需要折扣，只需对次数小的进行折扣计算，出现次数 *r* 为经验值，实践证明，当$r \le 5$，折扣系数$d_r = 1$时计算折扣是一个好的选择。从所有出现非 0 次的 *N* 元文法中折扣出去的总次数，等于赋给出现 0 次的所有 N 元文法的总次数

$$\sum_{W_1^N: c(w_1^N)>0} \left(P(w_1^N) - P_{Katz}(w_1^N)\right) = \sum_{0<r\le k} n_r(1-d_r)\frac{r}{N} = \frac{n_1}{N} \quad (3.85)$$

同时要保证d_r得到的折扣同 Good-Tuning 估计预测的折扣成一定比例关系，这个约束对应于下式，其中μ为常数

$$1 - d_r = \mu\left(1 - \frac{r^*}{r}\right) \quad (3.86)$$

从式(3. 85)和式(3. 86)可获得唯一解

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1-\frac{(k+1)n_{k+1}}{n_1}} = \frac{\frac{(r+1)n_{r+1}}{rn_r} - \frac{(k+1)n_{k+1}}{n_1}}{1-\frac{(k+1)n_{k+1}}{n_1}} \tag{3.87}$$

由此可以计算出每一个次数为 *r* 平滑后的值。实验证明，Katz 平滑在 2-gram 模型中具有较大优势。

（d）插值平滑技术

这类平滑方法直接利用模型中能提供的所有信息，通过归一化方法获得平滑后的概率值，此类平滑技术有线性插值平滑和非线性插值平滑两种。本书中，我们只介绍线性插值，非线性插值可根据兴趣作为课后扩展阅读。

线性插值平滑（linear interpolation smoothing）方法通常也称作 Jelinek-Mercer 平滑。它主要利用低阶模型对高阶 N-gram 模型进行线性插值。线性插值的平滑公式为

$$P_{interp}(w_i|w_{i-N+1}^{i-1}) = \lambda_{i-N+1}^{i-1} P_{ML}(w_i|w_{i-N+1}^{i-1}) + (1-\lambda_{i-N+1}^{i-1}) P_{interp}(w_i|w_{i-N+2}^{i-1}) \tag{3.88}$$

其中λ_{i-N+1}^{i-1}为插值系数。N-gram 模型可以递归地定义为由最大似然估计原则得到的 N-gram 模型和(N-1)-gram 模型的线性插值。当递归到 1-gram 时，可以令其为最大似然估计模型，或为一个均匀分布模型$P(w_i) = |V|^{-1}$。

对于插值系数λ_{i-N+1}^{i-1}的估计，一般可以用 Baum-Weltch 算法估计出来。其思想是：使用经过数据平滑的模型概率参数，计算一个测试集 T 的对数似然概率$\log P(T)$。当$\log P(T)$为极大值时，对应的λ_{i-N+1}^{i-1}为最优值。因此，令$\frac{\partial \log P(T)}{\partial \lambda_{i-N+1}^{i-1}} = 0$，通过求解该方程可得$\lambda_{i-N+1}^{i-1}$的迭代计算公式：

$$\hat{\lambda}_{i-N+1}^{i-1} = \frac{1}{c(w_{i-N+i}^{i-1})} \times \sum_{w_i} c(w_{i-N+1}^{i-1}) \times \frac{\lambda_{i-N+1}^{i-1} P_{ML}(w_i|w_{i-N+1}^{i-1})}{\lambda_{i-N+1}^{i-1} P_{ML}(w_i|w_{i-N+1}^{i-1}) + (1-\lambda_{i-N+1}^{i-1}) P_{interp}(w_i|w_{i-N+2}^{i-1})} \tag{3.89}$$

其中，$c(w_i)$是w_i出现的次数，$\hat{\lambda}_{i-N+1}^{i-1}$是本次迭代新的插值系数。

由于此类平滑技术的计算极其复杂，因此也就衍生出了很多改进的平滑算法，如 Wittem-Bell 平滑和 average-count 平滑。这两个平滑算法都是 Jelinek-Mercer 平滑的一个特例，与一般插值的不同在于λ_{i-N+1}^{i-1}的设置方式。

3.3.2 基于深度学习的语言模型

前面提到的基于 N-gram 的语言模型是传统方法，近年来，深度神经网络在分类等任务上具有很好的效果，人们开始研究基于神经网络的语言模型。2003 年 Bengio 等人提出了前向神经网络（Feed-forward Neural Network, FNN）[43]语言模型，该模型在当时具有较好的性能，吸引了大量的学术界和工业界的研究人员。2010 年 Mikolov 等人将循环神经网络（RNN）用于语言模型[44]，使得语言模型的性能得到了较大的提升，接着基于长短期记忆（Long Short Term Memory, LSTM）[45]循环神经网络的语言模型，进一步提升了模型的效果。RNN LM 利用上下文预测下一个单词，但并不是上下文中的所有单词都对下一个单词的预测有效，注意力机制可以从上下文的单词中选择有用的单词特征，更好的利用历史信息。2014 年 Bahdanau 等人[46]首次将注意力机制用于 NLP 任务，2016 年 Tran 等人和 Mei 等人证明了注意力机制可以提升 RNNLM 的性能。2018 年 OpenAi 提出了 GPT 模型（Generative Pre-Training）[47]，这是基于 transformer[48]（完全基于自注意力机制）模型进行无监督训练得到的，在 NLP 多个任务中的效果都超越了之前的模型。本部分主要介绍 FNN 语言模型和 RNN 语言模型。

3.3.2.1 FNN 语言模型

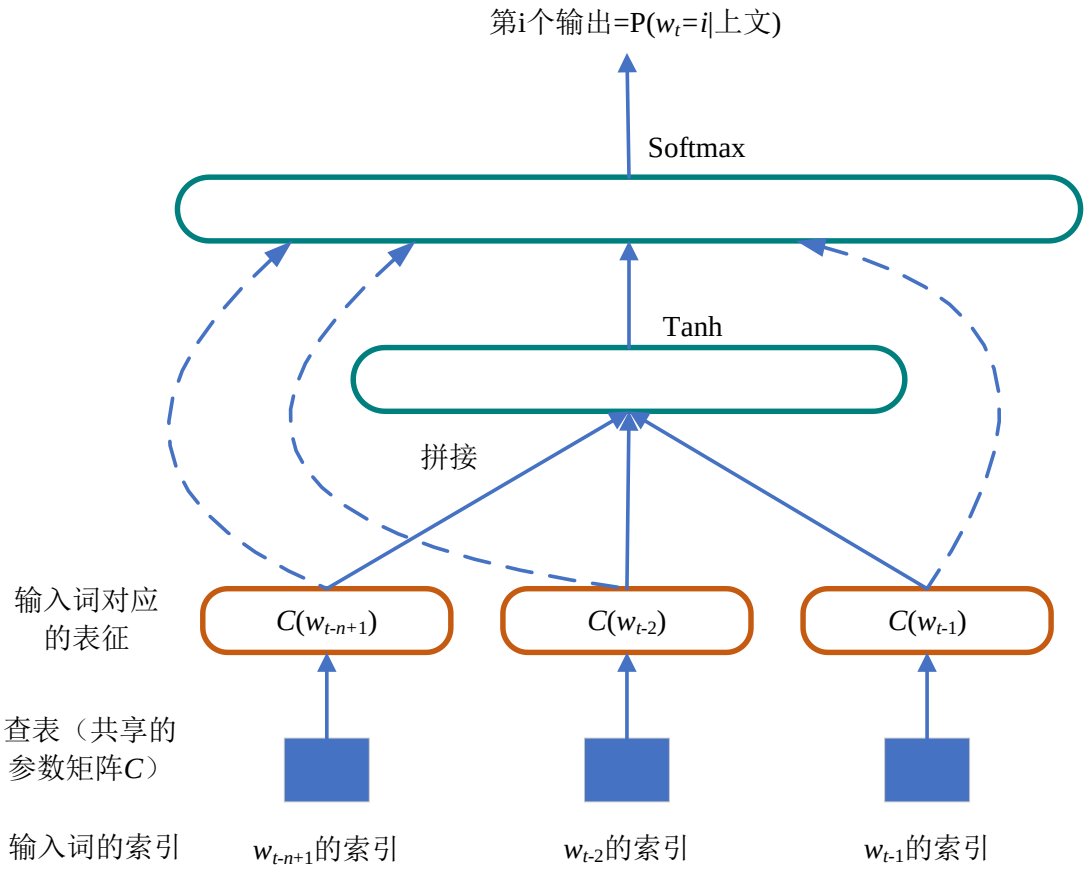


图 3-20 FNN 的网络结构：$f(i, w_{t-1}, \ldots . w_{t-n+1}) = g\big(i, C(w_{t-1}), \ldots, C(w_{t-n+1})\big)$，其中 g 是神经网络，C(i)是第 i 个单词的特征向量（引自[43]）

图 3-20 解释了 FNN 语言模型的过程，输入是一个词序列$\mathrm{w_{t-n+1}, w_{t-n+2}, \ldots, w_{t-1}}$，输出是预测的下一个词$\mathrm{w_i}$的概率，所以标签是$\mathrm{w_i}$，模型训练的目的是预测的概率分布接近 w_i 的真实分布。首先输入的词序列$\mathrm{w_{t-n+1}, w_{t-n+2}, \ldots, w_{t-1}}$是以 one-hot 向量表示的，它的维度等于词汇表的大小，记为$|V|$。Look-up 的作用就是将输入的词序列中的每个词通过一个全连接网络（参数共享）映射成低维向量（假设为m），然后将得到的低维向量拼接在一起，得到了 $(n-1)\times m$ 的向量，接着继续往前传播，遇到了激活函数为 tanh 的全连接网络（假设隐层有h个单元），输出h维的向量a，接着传播到最后的 softmax 层，注意 softmax 的输入包含了之前隐层的输入和输出，并不是只有隐层的输出，即 $\mathrm{P(w_i = i|context) = softmax(Wx + Ua + b)}$（W, U是权重矩阵，b是偏置向量）。损失函数是

$$\mathrm{L = \frac{1}{T}\sum_t \log \hat{P}\,(w_t|w_{t-1}, \ldots, w_{t-n+1}) + R(\theta)} \qquad (3.90)$$

其中$R(\theta)$是正则化项，通过反向传播即可训练。

3.3.2.2 RNN 语言模型

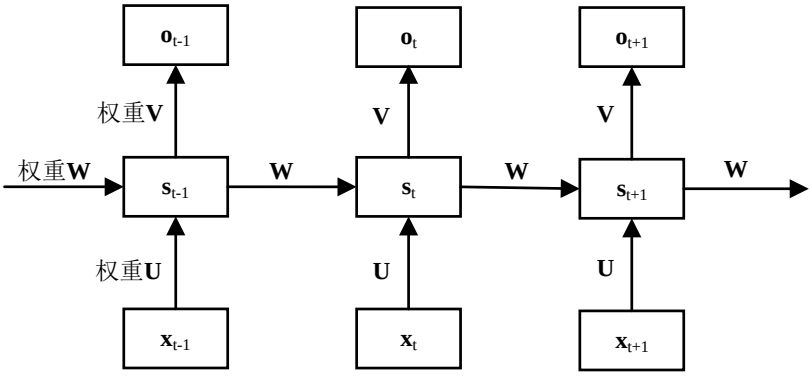


图 3- 21 RNN 的基本结构

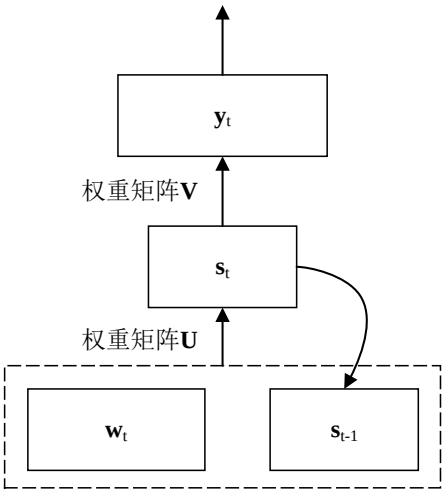


图 3- 22 RNN 语言模型

RNN（Recurrent Neural Network）是一类用于处理序列数据的神经网络[49]。结构示意图见图 3-21，与 FNN 不同的是，隐层向量$\mathbf{s_t}$的值不仅与输入$\mathbf{x_t}$有关，还与上一时刻的隐层值$\mathbf{s_{t-1}}$有关，可以用下述公式来描述 RNN：

$$\mathbf{o_t} = \mathrm{g}(\mathbf{V} \cdot \mathbf{s_t}) \tag{3.91}$$

$$\mathbf{s_t} = \mathrm{f}(\mathbf{U} \cdot \mathbf{x_t} + \mathbf{W} \cdot \mathbf{s_{t-1}}) \tag{3.92}$$

其中，f,g 为激活函数，如 sigmoid 函数，**U**,**V**,**W** 是可学习参数，并且在每个时刻是共

享的。

由于 RNN 能够处理时序问题，而语言模型也是一个时序任务，所以用 RNN 来建模语言模型是一个不错的选择，下面我们将介绍基于 RNN 的语言模型。

如图 3-22 所示，在第 t 步，RNN 语言模型可以写成如下形式：

$$\mathbf{x}_t = \left[\mathbf{w}_t^T;\ \mathbf{s}_{t-1}^T\right]^T \tag{3.93}$$

$$\mathbf{s}_t = f(\mathbf{U}\mathbf{x}_t + \mathbf{b}) \tag{3.94}$$

$$\mathbf{y}_t = g(\mathbf{V}\mathbf{s}_t + \mathbf{d}) \tag{3.95}$$

其中$\mathbf{U},\mathbf{V}$为权重矩阵，$\mathbf{b},\mathbf{d}$为偏置向量，f为 sigmoid 函数，g表示 softmax 函数，$\mathbf{s}_t$表示时间t的隐层状态，$\mathbf{w}_t$表示第t时刻输入的单词，第t时刻的输入$\mathbf{x}_t$由当前时刻输入的单词$\mathbf{w}_t$和上一时刻的隐层状态$\mathbf{s}_{t-1}$拼接组成，通过 RNN 的前向传播，最终通过 softmax 得到下一时刻预测的单词的概率分布$\mathbf{y}_t$。RNN LM 可以通过基于时间的反向传播算法（BPTT）或截断式 BPTT 算法来训练。

由于在 RNN 的训练过程中，可能会发生梯度消失或者爆炸，LSTM 缓解了这个问题，Sundermeyer 等人[50]将 LSTM 引入到了 LM 中，并且提出了 LSTM-RNN LM。另外在 NLP 领域，一些形式相似的词往往具有相同或相似的意思，比如英文中动词的各种分词。Mikolov 等人尝试用字符进行建模[51]，分别在 RNN LM 和 FFN LM 进行了探究。但是基于字符级别的模型性能较差，为了解决性能上的缺陷，Kim 等人提出了 CNN 与 LSTM 相结合的模型，其中 CNN 用于提取字符特征，然后输送给 LSTM，实验证明，这是一种有效的方法。Hwang 和 Sung 等人[52]使用一个分层 RNN 的结构进一步缓解字符级 RNN LM[53]的问题。近年来，transformer 模型在语音识别，机器翻译等领域取得很不错的效果。出现了不少基于 transformer 模型的语言模型，如 BERT[54]，GPT-2[55]等，这些模型借用超大规模的算力，并使用大语料进行训练，模型的效果超越了传统的算法结果，为下游的一些任务提供了较好的预训练模型。

在工业界，语音识别目前主要还是以传统的混合模型为主，并且由于训练时间、计算时间以及解码速度等原因，工业界语言模型还是以 N-gram 为主。

3.4 语音识别解码算法

3.4.1 基于 Viterbi 的解码算法

3.2.1 节介绍了 HMM 的三个基本问题，其中 Viterbi 算法解决的是解码问题，即给定一组观察序列，找出其对应的最优的 HMM 状态序列，Viterbi 算法本质上是一种动态规划算法，是可以得到全局最优解的。常见的基于 Viterbi 的解码算法有：token passing 算法[56]和 beam search 算法[57]。本节将介绍其他们的简单实现。

一个语音识别的解码网络是这样构建的，首先将发音词典中的单词取出并联，然后用对应的音素将其替换，接着再将音素换成更小的粒度——HMM 状态；最后把状态网络的首尾根据音素上下文一致的原则进行连接，构成回环。这样，一个语音识别网络最终由多个 HMM 状态组成。对于一段 T 帧的语音，经过 T 个 HMM 发射状态的路径都是一个可能预测结果。这样的每一条路径都有对应的概率值（一般会取 log，这样概率相乘就变成了相加），概率是由该条路径上每个独立的转移概率与发射状态的概率相加而成。HMM 状态之间的转移概率是由 HMM 的参数决定的，HMM 模型之间的转移概率是常数，HMM 结束状态的概率由语言模型决定。解码器的任务就是在这样的状态网络中找出概率最大的路径。

（1）Token passing 算法[56]

我们定义 HMM 模型为$\lambda = (\mathbf{A}, \mathbf{B}, \pi)$，隐藏状态为$\mathbf{Q} = \{q_1, q_2, \dots q_T\}$，观测状态为 $\mathbf{V}=\{v_1, v_2, \dots v_K\}$，我们的目标是给定观测状态，求对应的隐状态，使得整体的代价最小。定义$s_j(t)$是关于$\{v_1, v_2, \dots, v_t\}$与$\{q_1, q_2, \dots, q_j\}$的对齐的最大概率，那么我们可以按照如下的公式递归计算：

$$P_j(t) = \min_i\{s_i(t-1) + A_{ij}\} + B_j(v_t) \qquad (3.96)$$

所以整个模型的最大概率$P_{max} = \min_j P_j(T)$，其中j为任意一个 HMM 终止终止状态，T为观测状态的时间。解码算法用于解决上述问题。

Token passing 算法是这样求解最优路径的：让 HMM 上的每个状态都拥有一个可以移动的 token，用于保存部分解码路径和当前的概率$P_j(t)$。在时刻 0，token 被放置在每个可能的起始状态。在每个时间步，token 沿着转移路径传播，当遇到 HMM 的起始状态停止。当一个节点有多个出口时，这个 token 会被复制多份，然后并行的搜索可能的路径。当 token 在沿着转移路径传播时，它的概率由上述公式计算得到，并且记录路径。并且在每个状态中，比较所有 token，留下概率最高的 token，抛弃其他所有 token，这就是通过剪枝来加速解码。最终比较所有终止状态对应的 token，选择概率最大的 token，从而得到解码结果。具体算法如下：

初始化： 每个初始状态添加一个初始的 token，分数为 0，其他状态的分数为$-\infty$
算法： For t:=1→T: For 状态 i, do 复制若干数目 Token，并将其传递至所有与该状态连接的其他状态中，按照上述公式进行更新 token 的分数； End 丢弃原来的状态的 token For 状态 i, do 在每个状态中，留下分数最高的 token，抛弃其他所有 token； End End
结束： 对于所有的终止状态，找出分数最高的 token，返回解码结果。

（2）Viterbi beam search 算法[57]

原始的 Viterbi 算法是在整个搜索空间上搜索的，可以得到全局最优解，但是当搜索空间很大的时候，会存在很多条搜索路径，会很耗时，所以我们需要在 Viterbi 算法上应用剪枝算法，裁剪那些分数很低的路径，加速搜索，这就是 Viterbi beam search 算法的基本思想。下面我们详细介绍该算法。

在 Viterbi beam search 算法中有两个基本参数，$D(t;s_t;w)$和$H(t;s_t;w)$。$D(t;s_t;w)$表示时刻t到达词w的状态s_t的最优路径得分，$H(t;s_t;w)$记录时刻t到达词w的状态s_t的回溯指针。我们考虑词内和词间两种跳转。

词内跳转满足以下规则：

$$D(t;s_t;w)=\min_{s_{t-1}}\{A_{ij}+B_j(v_t)+D(t-1;s_{t-1};w)\} \tag{3.97}$$

$$H(t;s_t;w)=H(t-1;b_{\min}(t;s_t;w);w) \tag{3.98}$$

其中，A和B分别是 HMM 状态转移概率矩阵和发射概率矩阵，v_t是t时刻的观测向量。$B_{\min}(t;s_t;w)$表示计算出$D(t;\ s_t;w)$对应的s_{t-1}。

词间跳转满足以下规则：

$$D(t;\eta;w)=\min_{v}(\log P(w|v)+D(t;F(v);v)) \tag{3.99}$$

$$H(t;\eta;w)=<v_{\min},t>::H(t;F(v_{\min});v_{\min}) \tag{3.100}$$

其中，$F(v)$表示词v的终止状态，η表示伪起始状态；$P(w|w')$表示语言模型中的二元文法概率，$::$表示链接操作。此外，$v_{\min}=\arg\min_{v}\log P(w|v)+D(t;F(v);v)$。

假设我们搜索宽度为θ，那么当t时刻完成扩展后，我们计算t时刻时最优路径得分，记为Q，那么我们删除不满足以下条件的路径：

$$D(t;s_t;w)<Q-\theta \tag{3.101}$$

这样我们就可以通过调整θ的大小来平衡搜索速度和准确率。

Viterbi beam search 算法流程如下：

初始化：

$I(w)$表示词w的起始状态；

对所有可能是句子开始的词做以下操作：$D(0;I(w);w)=0$；$H(0;I(w);w)=\text{null}$

算法：

For t:=1→T

对于所有活动节点：

词内跳转更新D和H，按照公式：

$$D(t;s_t;w)=\min_{s_{t-1}}\{A_{ij}+B_j(v_t)+D(t-1;s_{t-1};w)\}$$

$$H(t;s_t;w)=H(t-1;b_{\min}(t;s_t;w);w))$$

对于所有活动词的终止状态，执行词间转移更新，按照公式

$$D(t;\eta;w)=\min_{v}(\log P(w|v)+D(t;F(v);v))$$

$$H(t;\eta;w)=<v_{\min},t>::H(t;F(v_{\min});v_{\min})$$

If $D(t;\eta;w)<D(t;I(w);w)$

$$D(t;I(w);w)=D(t;\eta;w);H(t;I(w);w)=H(t;\eta;w)$$

剪枝：找到最好的路径，按照设定的阈值，删除得分低的路径

结束：

在时刻T选出所有可能终止状态中的最好路径，并且使用$H(t;\eta;w)$回溯

3.4.2 基于加权有限状态自动机（WFST）的解码算法

3.4.2.1 WFST 概览

大词汇量连续语音识别（LVCSR）相比小词汇量的语音识别具有更多的困难，首先它拥有一个更大的词典，而且语音是连续的，很难确定单词的边界，另外加上了较大的语言模型使得解码的搜索空间更大了，所以找到一个精确的结果是不现实的。前面提到的 Viterbi 算法适合小词汇量的语音识别模型的解码，并且可以找到精确解，对于 LVCSR，解码具有更复杂的算法，另外，LVCSR 的搜索空间很大，解码过程需要被剪枝，这样才能更快的找到近似解。接下来将介绍 WFST 的基本概念以及基于 WFST 的语音识别解码算法（这是目前最常用的解码算法）。

3.4.2.1 WFST 的基本概念

（a）有限状态自动机（Finite State Automaton，FSA）

有限状态自动机是定义状态机的模型，该状态机响应于某些输入而在状态之间进行转换。如图 3-23 所示，总共有 4 个状态，每条弧上的字符代表输入，例如，当前在 S1 状态，当接收了输入 c，那么它就转移到了 S2。

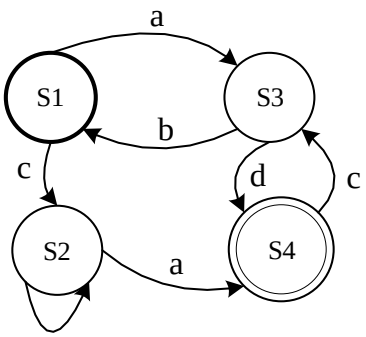


图 3- 23 有限状态自动机（FSA）示意图

（b）加权有限状态接收器（Weighted Finite State Acceptor，WFSA）

有限状态接受器是一组可能无限的标签的有限表示。它将标签序列与初始状态和最终状态之间的路径匹配。 如果找到了这样的路径，则接受该标签序列，否则，将被拒绝。再在每个弧上添加权重就是 WFSA 了，这样不仅可以判断某个输入是否能被接收，还能计算出其权重，如图 3-24 所示。当弧上没有输入（一般用$\mathcal{E}$或<eps>表示），这就表明任何输入都能走这条路径。例如，在图 3-24 中，该 WFSA 将会接受输入序列“bcca”，而将拒绝“abc”。

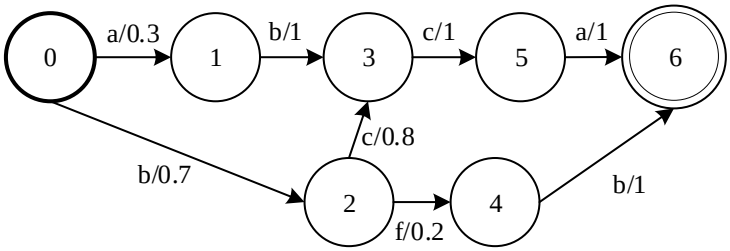


图 3- 24 WFSA 示意图

（c）加权有限状态转换器（Weighted Finite-State Transducers，WFST）

Finite-State Transducers（FST）与 FSA 的区别是它每条弧上不仅有输入，还有对应的输出。对于图 3-25，若输入序列为“abd”，那么该 WFST 可以对此进行匹配，然后将第一条弧上的“A”输出。ε 表示空。

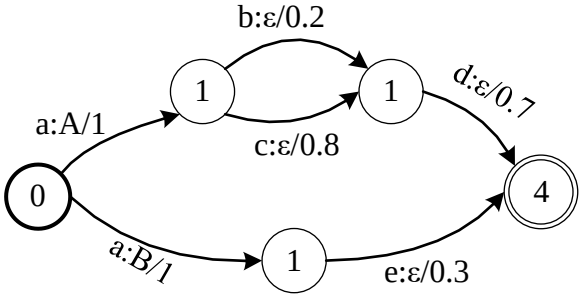


图 3- 25 WFST 示意图

3.4.2.2 基于 WFST 的语音识别

一个 HMM 模型就是一个状态转换机，它表示一个音素对应的可能的 HMM 状态序列，

如图 3-26 所示。发音词典也可以转换成一个状态转换机，图 3-27 说明了单词“found”是如何发音的，以及可能存在的发音情况。例如，音素/f/后面可以接/aa/或者/aw/。图 3-28 是单词“Found”发音词典对应的 WFST 图。此外，语言模型也可以变成一个状态转换机，图 3-29 是一个二元语言模型（bigram），即当前词出现的概率只由它前面的词决定的。图 3-29 中有两个句子，“i found him at home”和“i am at his house”。图 3-30 是语言模型对应的 WFST 图，其输入和输出是一样的，并且每条弧上都有权重，这样就可以很方便计算某个输入的句子的概率。

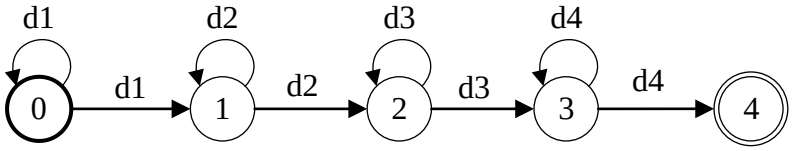


图 3- 26 HMM 模型[58]

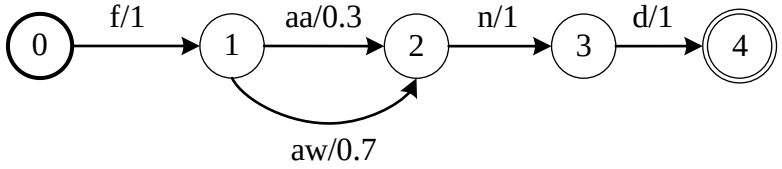


图 3- 27 发音词典（“Found”）

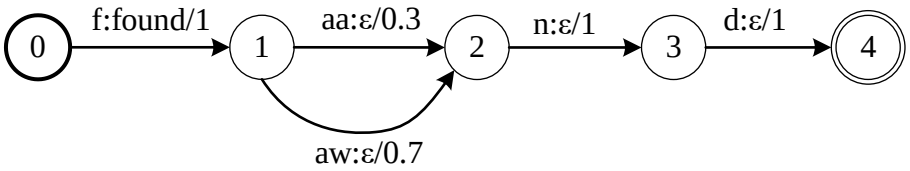


图 3- 28 发音词典 WFST 示意图（“Found”）

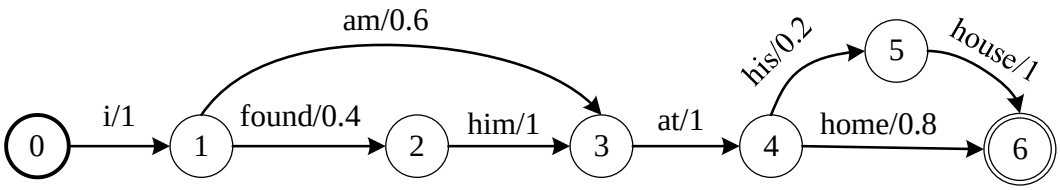


图 3- 29 语言模型

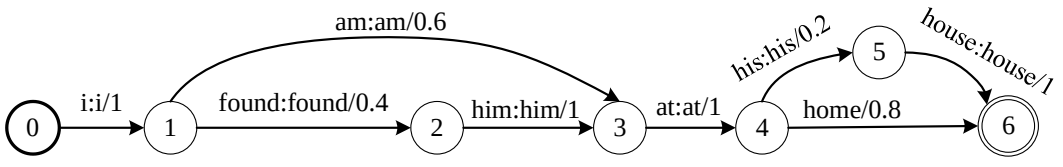


图 3- 30 语言模型 WFST 示意图

将这些状态转换机进行扩展，使之给定一个输入序列可以得到对应的输出序列，这就是 WFST。基于 WFST 的解码中有四种 WFST，HMM 转换机（H），上下文相关（context-dependency）转换机（C），发音词典（L），语言模型（G）。HMM 转换机将 HMM 的状态映射到上下文相关的音素（CD-phones），上下文相关转换机将 CD-phones 转化成上下文无关的音素（CI-phones），这样我们才能把它们映射成单词。发音词典和语言模型转换机将音素转成单词，又将单词转成通顺序的句子，即最终的识别结果。在语音识别解码中，我们将这四种 WFST 组成一个大的解码图（$H \circ C \circ L \circ G$），然后在这个解码图上进行搜索，将 HMM 状态转换成最终的句子。理论上我们可以使用 Viterbi 算法进行搜索，但是由于这个图十分庞大，在实际中我们会对这个图进行简化，以获得更快的解码速度。对这个解码图的优化，主要有以下几个操作，确定化（det），最小化（min），权重推移（weight-pushing），这将在 3.4.2.3 节中进行介绍。

$$HCLG = \min(\det(H \circ \min(\det(C \circ \min(\det(L \circ G)))))) \tag{3.102}$$

以 $\min(\det(L \circ G))$ 为例，如图 3-31，输入“/b/ /ih/ /l/ /#0//f/ /l/ /eh/ /d/ /#0/”，可以得到输出“Bill fled”；输入“/jh/ /ih/ /m/ /#0/ /r/ /eh/ /d/ /#0/”可以得到输出“Jim read”。但是实际中的解码图比这个更大更复杂，所以一般在使用 Viterbi 解码的时候会加上束剪枝（beam-pruning），即在搜索过程中，当某些路径的代价已经达到了预定义的阈值，就可以把这条路径丢弃掉，或者每次搜索后，只留下同样数量的最佳路径，这样可以大大的降低搜索的时间，并且精度上的损失也在接收范围内。

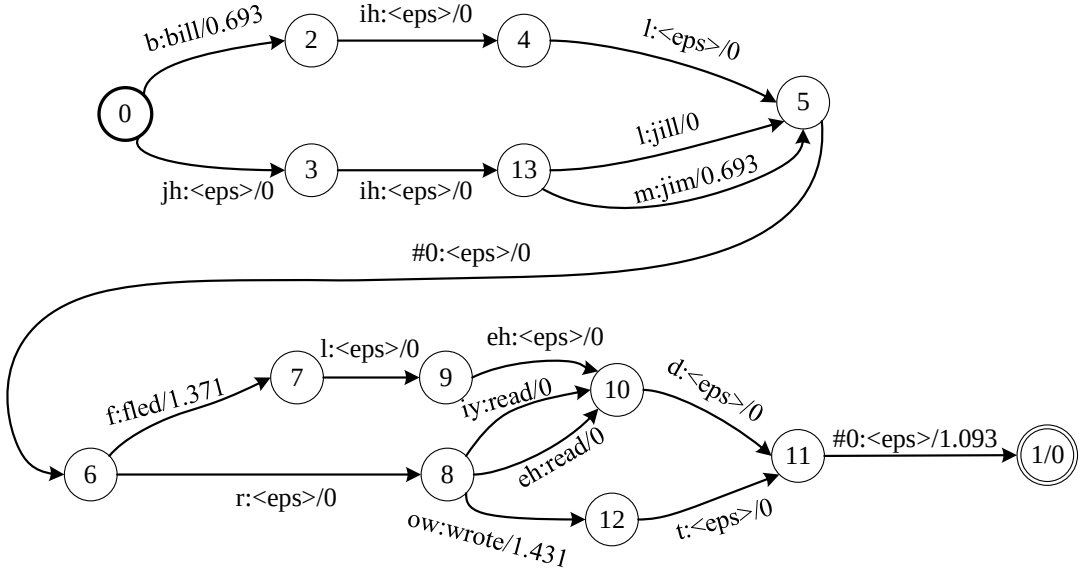


图 3- 31 min(det(L◦G))示意图（输出“Bill fled”或“Jim read”）[58]

3.4.2.3 WFST 优化算法介绍

（a）组合（Composition）

组合操作可以将不同的 WFST 合并成一个大的 WFST，这是构建整个解码图的第一步。在组合操作中，我们找出所有的弧组合，如果第一个 WFST 的某条路径的输出和第二个 WFST 的输入匹配，那么就创建一个新的弧，将这两条路径进行合并，同时权重相加。图 3-32(a)中的输入为“b b”的路径对应的输出是“a b”，在图 3-10(b)中找不到可以匹配的路

径，所以丢弃。

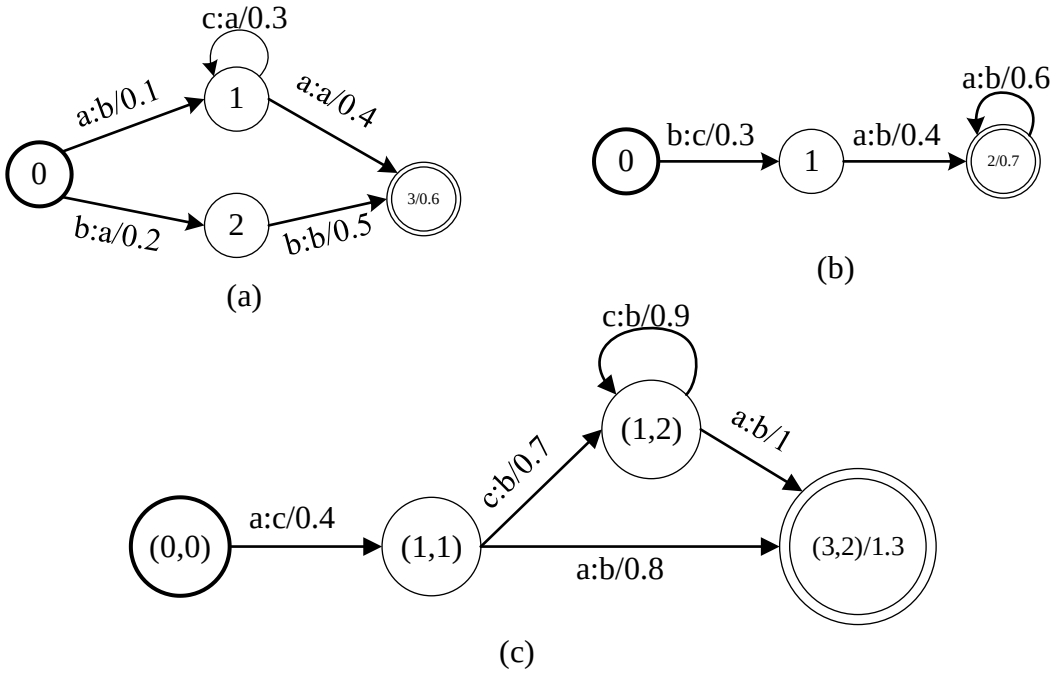


图 3- 32 组合操作示意图

（b）确定化（Determinization）

由于合并后的 WFST 比较庞大，所以为了提高搜索效率，对 WFST 中的路径进行确定化，如图 3-33。确定化后的 WFST 图从每一个状态出发的弧中没有相同的输入。即从每一个状态出发，给定一个输入，下一步的走向是唯一确定的。确定化的操作可以大大降低解码的时间，但是并不是所有的 WFST 都可以确定化。

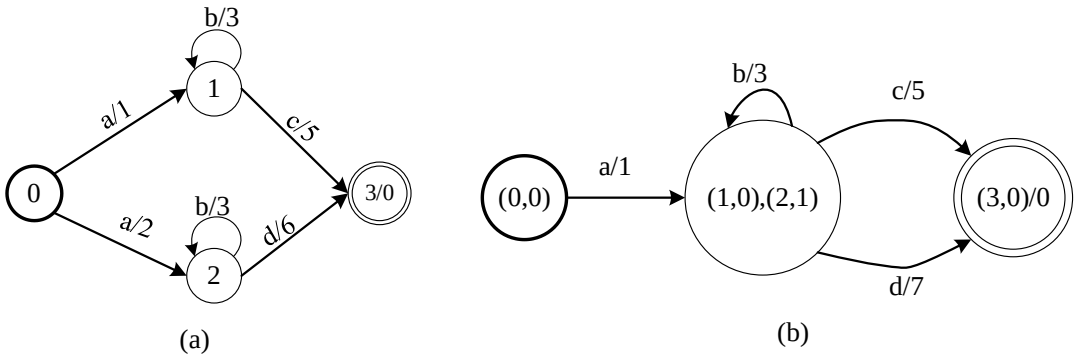


图 3- 33 确定化操作示意图。(b)为(a)确定化的结果

（c）权重推移（Weight-pushing）

WFST 中的权重可以被推向初始或者最终状态，在权重推移之后，权重可以重新在靠近头部分配，图 3-34(b)为图 3-34(a)权重推移的结果。权重推移的操作可以使剪枝算法更加高效，因为可以更早的看见权重分配。

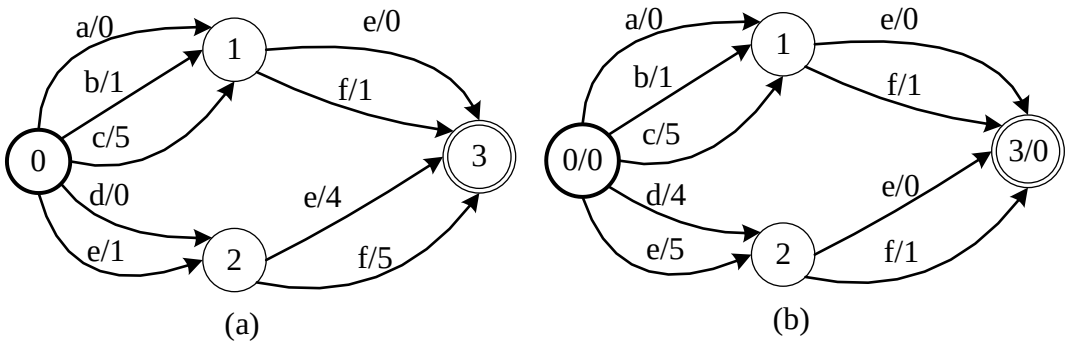


图 3- 34 权重推移示意图。(b)为(a)权重推移的结果[58]

（d）最小化（Minimization）

最小化的作用是减少 WFST 的状态，因为 WFST 中会存在一些冗余的节点，可以通过合并这些重复的节点达到简化 WFST 的效果，以提升解码速度。WFST 最小化操作可以使用经典的最小化算法，如 Hopcroft 算法。图 3-35 是一个最小化的例子，可以发现最小化的结果并不是唯一的，但是最小化的结果都具有相同的拓扑结构，只是路径上权重分布不一样。

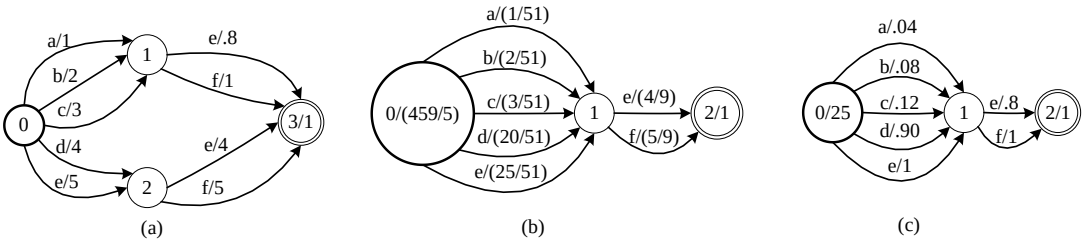


图 3- 35 最小化示意图。(b)和(c)都是(a)最小化的结果[58]

3.5 语音识别技术的展望

语音识别发展至今，为人们的生活带来了巨大的便利，虽然语音识别技术已经应用于人类生活中的各种场景中，并且在安静环境下识别率能达到 98%之高，被誉为“超过了人类”。然而，到目前为止，仍然有许多问题需要解决。下面简要介绍目前语音识别技术所面临的问题和挑战。

1. 复杂声学场景下的语音识别

当语音识别系统处于复杂声学场景下，由于训练声学模型的声学场景与真实复杂声学场景差异过大，导致语音识别系统性能大大降低。而现实直接中，声学场景千变万化，训练集不能完全覆盖所有声学场景，因此，复杂声学场景下的语音识别系统一直是语音领域中极具挑战性的问题。

复杂场景下的语音识别主要有以下难点：

- 真实世界传感器需要获取声音，但是周围声音是各种各样的，需要获取有效的人声
- 获取人声也不一定是目标任务的人声，可能是用户和别人交谈的声音
- 在远场场景下获取的声音面临着回声干扰、室内混响、多信号源干扰以及非平稳

噪声的干扰等等

面对这些复杂问题，当前最好的识别系统也无法获得很好的识别结果，因此需要研究新的声学信号处理技术或新的声学信号处理与语音识别的联合优化方案来解决，以提高语音识别的性能。

2. 低数据资源语音识别

低数据资源是指音频或者有转录文本的音频较少，导致语音识别性能下降，该问题是当今语音识别研究领域的热点问题之一。全世界共使用 5651 种语言，其中，汉、英、印度、俄、西班牙、德、日、法、印度尼西亚、葡萄牙、孟加拉、意大利和阿拉伯语是使用人数较多的语言，而其他一些少数人使用的语言在语音资源上就显得匮乏，这些资源匮乏的语言也许没有足够的训练数据来训练出一个好的语音识别系统，因此，低资源问题对于小语种语音识别是一个难点。

除了小语种的识别外，当一个表现良好的语音识别系统遇到一些人群时，性能也会大大下降，包括但不限于以下人群：

- 非母语学习者
- 构音障碍患者（由于疾病导致发音的错误，产生的语音可能扭曲、错误）
- 少数民族人群

这些人群的语音数据都由于说话人数少、发音困难等原因而较难获取到。为了让语音识别系统能够尽可能为所有人，而不是大多数人带来便利，低资源问题是必须解决的一个难点问题之一，也是一个非常具有研究价值的问题。目前，基于迁移学习、模型自适应等算法的低数据资源语音识别受到广泛的关注。

3. 多语言和跨语言语音识别

目前大多数语音识别系统都是针对一种语言的，如果我们对于每种语言都训练一个语音识别系统，所需要的成本太高，且需要大量的数据。因此，多语言和跨语言语音识别正是研究如何只使用一个语音识别系统识别多种语言。多语言语音识别和跨语言语音识别的区别在于：多语言语音识别只能识别训练集中出现过的语言，而跨语言语音识别能识别出从未见过的语言。目前该研究所面临的问题是：

- 如何选择建模单元。像字符、子词等单元很难扩展到词汇量大的语言
- 词汇量大导致标签稀疏问题
- 构建发音字典需要每个语言专家的知识，要耗费很大的人力
- 有些语言很难获取

目前多语言和跨语言语音识别的准确率离实际应用还有很大的差距，这些问题使得研究变得困难，需要新技术、新的解决方案才能有新的进展。

4. 语种混杂语音识别

在我们日常交流中，经常会有中文语境下英文单词夹杂的现象，如，“这里有 Wi-Fi 吗”。这在学术上称为语种混杂（Code-switch）。我们知道，汉语是世界上使用人数最多的语言，而英语是使用范围最广的语言。近年来，随着中国外交不断深入，中英混说的现象在我们生活中也越来越普遍，而目前中英混合的识别率并不太好，因此，该研究成为了当前语音识别技术面临的重要挑战之一。目前该研究需要解决的问题包括但不限于：

- 由于嵌入语（英文）受主体语（中文）影响形成的非母语口音现象严重
- 不同语言音素构成之间的差异给混合声学建模带来巨大困难
- 带标注的混合语音训练数据极其稀缺

最近，基于端到端的语音识别算法与技术逐步被应用到语种混杂语音识别任务上。

5. 多口音语音识别

在日常生活中，一个地区的人在说另一个地区的语言时，容易保持自己习惯的发音方式，因此，会出现不同的口音。以汉语为例，汉语中共有八大方言，即官话、吴语、湘语、赣语、客家语、闽南语、闽北语以及粤语，其中，官话是与标准普通话最为接近的一种方言，其他各种方言在声学发音以及语言学表现上都与标准普通话有着显著的差异。由于多数普通话使用者把普通话作为第二语言来掌握，他们的普通话发音不可避免地受到其方言母语发音的强烈影响，出现发音不准确、发音错误等现象，导致语音识别性能下降。简而言之，目前多口音语音识别存在的问题大致有以下两点：

- 受母语影响不同方言背景的说话人的发音具有差异性
- 可用于训练语音识别系统的多口音的数据极其稀缺

对于第一个问题，虽然不同方言口音之间存在着差异，但也存在着相似性，因此，研究如何改进声学模型以适应更多变化的口音是一种研究趋势。第二个问题也是现在语音识别的难点之一。

6. 总结

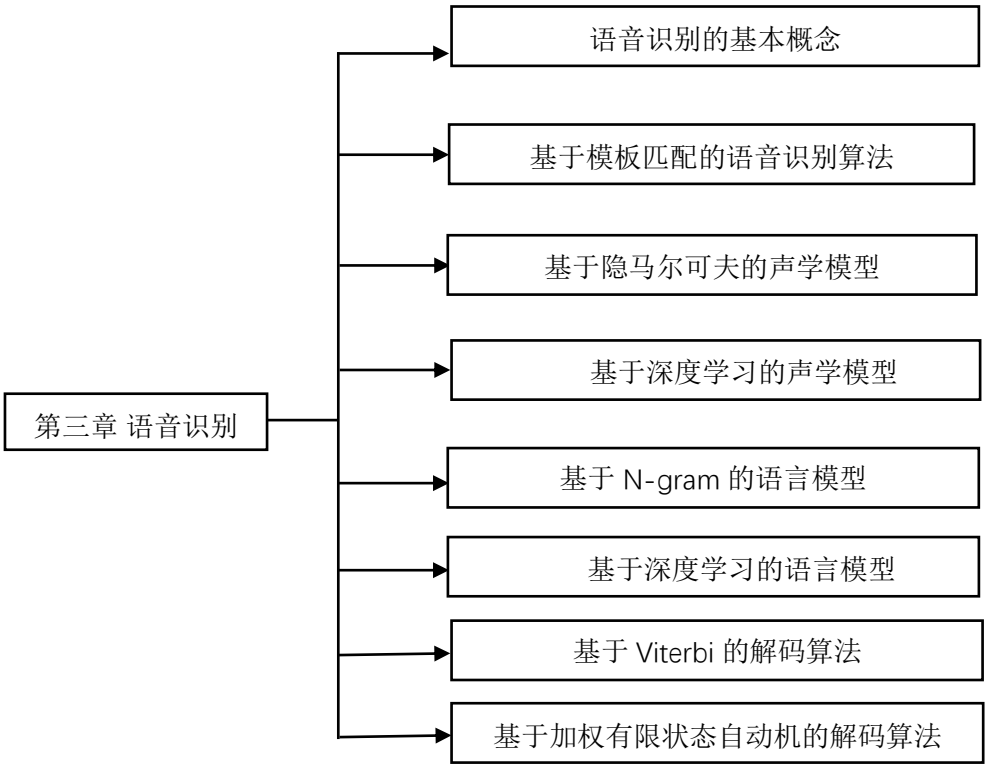
从上述介绍中，我们可以看出，语音识别存在的问题有着一定的相似性，当我们解决了其中一个问题，其他任务的问题也随之解开。除了以上的技术难点，随着用户对语音中的个人隐私数据的关心，最近基于低计算资源的设备端、边缘端的语音识别研究也越来越受重视。我们相信，在未来语音识别技术能有更大的突破，这些困难也能得到解决，而从中受益的不仅仅是我们这些普通人，更是那些无法从当今人工智能技术中得到便利或帮助的人们。

3.6 小结

在本章，我们从语音识别应用出发，介绍了语音识别的分类、语音识别的发展历史以及早期语音识别系统的实现方法。进一步，我们分别从语音识别的三个重要模块进行介绍：声学模型、语言模型、解码算法。本章用了较大的篇幅介绍了声学模型，介绍了基于传统统计学方法的声学建模以及基于深度学习的声学建模，并讨论了几种有潜力的声学模型。无论是统计学方法还是深度学习方法都在目前的语音识别系统的研究和应用中有着不可忽视的作用。随后，我们介绍了两种常用的语言模型：基于 N-gram 的语言模型和基于深度学习的语言模型，这些方法由于性能优良，在学术界及工业界都受到广泛的关注。接着，我们介绍了语音识别的解码算法：基于 Viterbi 的解码算法以及基于加权有限状态自动机的解码算法。这些算法在语音识别系统中发挥了巨大的作用。本章的最后，介绍了语音识别技术目前面临的挑战。

语音识别技术虽已成功应用于我们日常生活中的多种应用，其性能也达到广大用户的人课，但仍然存在许多挑战。研究者们还在向让语音识别系统更快、更准地识别语音的目标努力着。

本章知识点小结



3.7 语音识别实践

前面介绍了语音识别的基础理论知识，本节将结合理论，实现语音识别系统搭建。首先介绍可用于语音识别任务的公开免费数据集，以供读者选择实验数据，然后介绍一些开源的语音处理工具，最后介绍如何利用公开数据集及语音处理工具搭建语音识别系统。

3.7.1 开源数据集

工欲善其事必先利其器，做机器学习，我们需要有利器，才能完成工作，数据就是我们最重要的利器之一。做语音识别，我们需要有对应的语音数据集，以帮助我们完成和不断优化改进项目。我们可能很难拿到成千上万小时的语音数据集，但是这里有一些免费开源的语音数据集，可以提供给大家做一些基础实验。

3.7.1.1 英文数据集

（a） Librispeech

该数据集是包含大约 1000 小时的英语语音的大型语料库。这些数据来自 LibriVox 项目的有声读物。它已被分割并正确对齐，该数据集在 kaldi-asr.org 上有训练好的声学模型和语言模型，适合用于英文的语音识别模型预训练。

下载地址：http://openslr.org/12/

（b）Aurora4

该数据集是鲁棒语音识别的研究，为了比较不同前端在大词汇任务中的识别性能而创建的。与 Aurora-2 数据集一样，噪音信号被人为地添加到相当干净的数据中。共含有采样率为 8-kHz 和 16-kHz 的版本。

下载地址：http://aurora.hsnr.de/aurora-4.html

（c）AMI

AMI 会议语料库是一个包含 100 小时会议记录的多模式数据集。有关语料库的简要介绍，请参阅语料库概述及文档。大约三分之二的数据是通过一个场景引出的，在这个场景中，参与者在一个设计团队中扮演不同的角色，在一天的时间里从开始到完成一个设计项目。其余部分由一系列领域中自然发生的会议组成。该数据库所用的同步录音设备包括：近距离交谈和远场麦克风，个人和房间视图摄像机，投影，白板，个人笔。音频所对应的注释包括：转录文本和许多不同现象的注解（对话行为，头部运动等）。该数据库还可广泛应用于除语音识别的其他领域。

下载地址：http://openslr.org/16/

（d）TED-LIUM

TED- lium 语料包含英语 TED 演讲及其转录文本，音频的采样率为 16kHz。它包含了大约 118 个小时的演讲。

详细介绍可参考以下文章：

A. Rousseau, P. Deléglise, and Y. Estève, "TED-LIUM: an automatic speech recognition dedicated corpus", in Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12), May 2012.

下载地址：http://openslr.org/19/

3.7.1.2 中文数据集

（a）THCHS30

THCHS30 是一个很经典的中文语音数据集了，包含了 1 万余条语音文件，大约 40 小时的中文语音数据，内容以文章诗句为主，全部为女声。它是由清华大学语音与语言技术中心（CSLT）出版的开放式中文语音数据库。原创录音于 2002 年由朱晓燕教授在清华大学计算机科学系智能与系统重点实验室监督下进行，原名为“TCMSD”，代表“清华连续”普通话语音数据库’。13 年后的出版由王东博士发起，并得到了朱晓燕教授的支持。他们希望为语音识别领域的新入门的研究人员提供玩具级别的数据库，因此，数据库对学术用户完全免费。

下载地址：http://openslr.org/18/

（b）AIShell-1

希尔贝壳中文普通话开源语音数据库 AISHELL-ASR0009-OS1 录音时长 178 小时，是希尔贝壳中文普通话语音数据库 AISHELL-ASR0009 的一部分。AISHELL-ASR0009 录音文本涉及智能家居、无人驾驶、工业生产等 11 个领域。录制过程在安静室内环境中， 同时使用 3 种不同设备： 高保真麦克风（44.1kHz，16-bit）；Android 系统手机（16kHz，16-bit）；iOS 系统手机（16kHz，16-bit）。高保真麦克风录制的音频降采样为 16kHz，用于制作 AISHELL-ASR0009-OS1。400 名来自中国不同口音区域的发言人参与录制。经过专业语音校对人员转写标注，并通过严格质量检验，此数据库文本正确率在 95%以上。分为训练集、

开发集、测试集。

该数据集总共 178 小时，400 个说话人，其中训练集有 340 个人，测试集有 20 个人，验证集有 40 个人，每个人大概录三百多句话，每个人讲的话都放在一个文件夹里面。

下载地址：http://openslr.org/33/

（c）AIShell-2

希尔贝壳中文普通话语音数据库 AISHELL-2 的语音时长为 1000 小时，其中 718 小时来自 AISHELL-ASR0009-[ZH-CN]，282 小时来自 AISHELL-ASR0010-[ZH-CN]。录音文本涉及唤醒词、语音控制词、智能家居、无人驾驶、工业生产等 12 个领域。录制过程在安静室内环境中， 同时使用 3 种不同设备： 高保真麦克风（44.1kHz，16bit）；Android 系统手机（16kHz，16bit）；iOS 系统手机（16kHz，16bit）。AISHELL-2 采用 iOS 系统手机录制的语音数据。1991 名来自中国不同口音区域的发言人参与录制。经过专业语音校对人员转写标注，并通过严格质量检验，此数据库文本正确率在 96%以上。

AISHELL-2 的结构和 AIShell-1 的类似，共有 1991 个说话人，每个人有 500 句话，每个人说的话可能会有重复。

下载地址：http://www.aishelltech.com/aishell_2

（d）aidatatang_200zh

Aidatatang_200zh 是由北京数据科技有限公司（数据堂）提供的开放式中文普通话电话语音库。

语料库长达 200 小时，由 Android 系统手机（16kHz，16 位）和 iOS 系统手机（16kHz，16 位）记录。邀请来自中国不同重点区域的 600 名演讲者参加录音，录音是在安静的室内环境或环境中进行，其中包含不影响语音识别的背景噪音。参与者的性别和年龄均匀分布。语料库的语言材料是设计为音素均衡的口语句子。每个句子的手动转录准确率大于 98％。

下载地址：http://openslr.org/62/

（e）MAGICDATA

Magic Data 技术有限公司的语料库，语料库包含 755 小时的语音数据，其主要是移动终端的录音数据。邀请来自中国不同重点区域的 1080 名演讲者参与录制。句子转录准确率高于 98％。录音在安静的室内环境中进行。数据库分为训练集，验证集和测试集，比例为 51：1：2。诸如语音数据编码和说话者信息的细节信息被保存在元数据文件中。录音文本领域多样化，包括互动问答，音乐搜索，SNS 信息，家庭指挥和控制等。还提供了分段的成绩单。该语料库旨在支持语音识别，机器翻译，说话人识别和其他语音相关领域的研究人员。因此，语料库完全免费用于学术用途。

下载地址：http://openslr.org/68/

3.7.2 语音识别工具介绍

3.7.2.1 HTK

HTK（Hidden Markov Model Toolkit）[59]是一个用 C 语言编写来搭建隐马尔可夫模型（HMM）的工具包，主要用于进行基于 HMM 的语音处理，特别是语音识别。由剑桥大学工程系（CUED）机器智能实验室开发维护。1989 年 HTK 发布了 1.0 版，开始只作为商业工具需要购买或者取得许可才能使用，直到 2000 年 9 月 HTK 开源免费使用

（http://htk.eng.cam.ac.uk/），其中包括 HTK 工具包和对应的工具书 HTKBook，需要在官网进行注册后才能下载。HTK 的最近一次更新时在 2016 年，加入了神经网络模块，发布了 3.5 测试版。

3.7.2.2 Kaldi

Kaldi[60]同 HTK 一样也是一个用来搭建隐马尔可夫模型（HMM）的工具包，架构如图 3-36 所示，底层是线性代数库 BLAS、LAPACK 和构造加权有限状态机的 OpenFST 库，通过 C++编写矩阵运算、构造 GMM、线性变换搭建语言模型、决策树、HMM、基于 WFST 的解码器工具等，并采用 bash 和 python 脚本进行进一步封装，方便使用者进行调用。kaldi 起源于 2009 年的约翰霍普金斯大学夏季研讨会，之后主要由 Daniel Povey 来进行开发维护。kaldi 可以用来做语音识别、说话人识别、语种识别、手写识别、OCR、图像分类，并且针对每类任务都有多个 recipes，方便初学者进行项目搭建，其中最主要的是语音识别。

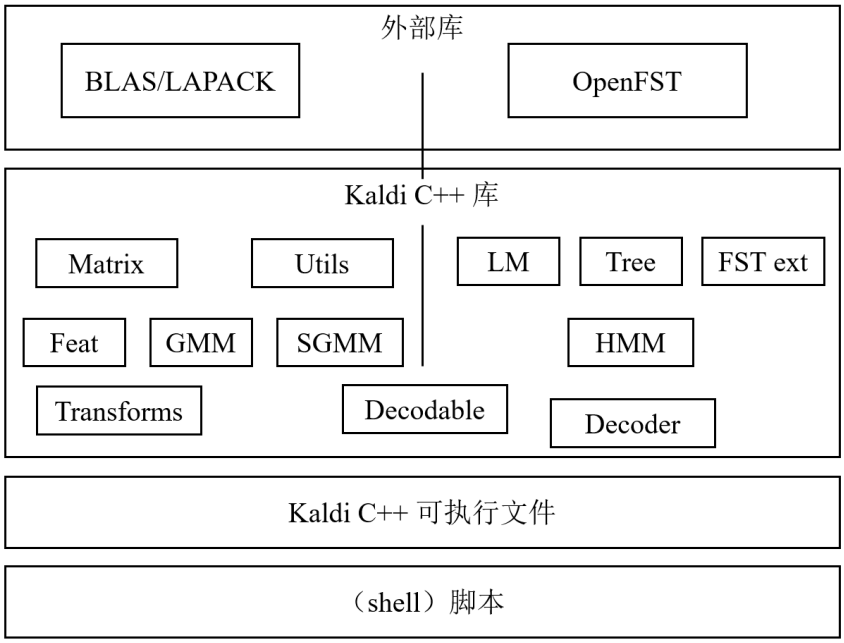


图 3-36 Kaldi 结构简图[59]

Kaldi 是一个持续发展的开源项目，最初是在 Sourceforge 上托管，现已经完全迁移到 github 上[61]，在 2017 年 1 月 5 日开始引入版本机制后，现已从最初的 5.0.0 版本更新到目前最近的 5.5.636 版，相比于 HTK，kaldi 更新更频繁，最近的一次大的版本更新是在 2018 年 8 月 24 日，其发布了 5.5 版，最近的一次小更新是在 2020 年 2 月 8 日其修复了一个 Bug。

Kaldi 从开始研发就汲取其他语音识别工具的经验，相比于 HTK，kaldi 有如下优点：采用加权有限状态机（WFST）进行解码、具有广泛的线性代数支持、可扩展性强、采用最新的限制最少的 Apache 2.0 开源协议、具有完整且能达到 SOTA 的示例脚本，这些都大大降低了语音识别的技术门槛，同时吸引了大量国内外用户，极大推动了语音技术的发展，现已成为主流的语音技术工具。目前示例如 aishell、WSJ、Switchboard、Librispeech、TED、HKUST、Voxforge 等，基于这些示例，开发者可模仿示例数据存放格式并快速搭建自己的语音识别系统，目前工业界基于 HMM 的语音识别系统基本都使用 kaldi 来进行搭建。

目前针对 kaldi 比较官方的文档资料有 kaldi 文档[62]和 2020 年 4 月新出的《Kaldi 语音

识别技术实战》[63]，分别为英文和中文供大家深入学习参考。

3.7.2.3 ESPNET

ESPNET[64]是一个使用 Chainer 和 Pytorch 作为深度学习引擎的端到端语音工具包，在运行时可以选择其中之一，ESPNET 可以用来进行端到端语音识别和语音合成。主要特点如下：可以使用 CTC、Attention 或者混合 CTC/Attention 来进行端到端语音识别；编码器采用 CNN+BLTM、金字塔状 BLSTM 或 Transformer Encoder，注意力机制可采用点乘注意力机制、位置感知注意力机制或者多头注意力机制；可以加入 RNNLM 或 LSTMLM 作为语言模型；数据处理、特征提取、保存格式和 recipes 借鉴 kaldi 的风格，并且在日语和中文语音识别上可以实现 SOTA，在英文语音识别也可以取得不错的表现。ESPNET 同样也是放在 github 上进行托管[64]。

3.7.3 实践——搭建语音识别系统

下面介绍如何使用 kaldi 工具搭建语音识别系统。本实验以 AIshell-1 数据集为例，依赖于 linux 环境。

3.7.3.1 Kaldi 工具安装

运行环境：centos7、ubuntu16.4、macos 10.13 等。

需安装的软件：apt-get、subversion、automake、autoconf、libtool、g++、zlib、libatal、wget 等。

kaldi 依赖包：

- OpenFST：Weighted Finite State Transducer library，是一个用来构造有限状态自动机的库。是一个构造、合并、优化和搜索加权有限状态机 FST 的库。隐马尔科夫模型可以看成是一个有限状态自动机，Kaldi 的文档里有这样一句话：If you ever want to understand Kaldi deeply you will need to understand OpenFst. 因此，这是 kaldi 工具最重要的一个依赖包。
- ATLAS：这是一个 C++下的线性代数库。许多矩阵运算都依赖于此包。
- IRSTLM：这是一个统计语言模型的工具包。
- sph2pipe：这是宾夕法尼亚大学 linguistic data consortium（LDC）开发的一款处理 SPHERE_formatted 数字音频文件的软件，它可以将 LDC 的 sph 格式的文件转换成其它格式。
- SCTK：Speech Recognition Scoring Toolkit 是 NIST（National Institute of Standards and Technology, 美国国家标准与技术协会）提出的一套工具集。NIST 评分工具包。
- SRILM：SRILM - The SRI Language Modeling Toolkit 是由 SRI International 提出的一套工具集，主要用于创建和使用统计语言模型。

实验中涉及到的 kaldi 工具的依赖环境需要读者自行安装配置，下面介绍在依赖环境搭建成功后，如何安装编译 kaldi 工具。

①首先，在 kaldi 官网用 git clone 命令将 kaldi 项目下载至本地：

~$ git clone https://github.com/kaldi-asr/kaldi

②然后，进入 kaldi 目录下的 tools 目录，编译 tools 目录。

~/kaldi/tools$ cd kaldi/tools

进入 extras 运行脚本 check_dependencies.sh 来检查各种依赖是否安装：

~/kaldi/tools/extras$ ./check_dependencies.sh

运行 check_dependencies.sh 后出现任何提示表明某些库未安装，都应按照提示解决，

直到运行 check_dependencies.sh 后出现如上所示”./check_dependencies.sh: all OK.”。

回到 tools 目录，进行编译：

~/kaldi/tools/extras$ cd ../

~/kaldi/tools$ make

如果是在虚拟机上，建议使用 make 而非 make -j 4，否则很容易内存不够导致编译失败，之后在 src 目录下的编译也一样。

make 完成后可能会提示 irstlm 未安装，此时不用管，先继续完成整个 kaldi 的安装再说。

③进入 kaldi 根目录下的 src 目录，编译 src 目录。

~/kaldi/tools$ cd ../src

运行 configure 且不要添加参数“--shared”：

~/kaldi/src$ ./configure

务必仔细阅读运行 configure 后显示的提示，它可能和上文所示的内容有所区别，其中提醒了你有哪些东西没安装好，并给出了指导，遵循那些执导完成相关依赖的安装，直到运行 configure 后出现如上文所示的提示，提示的最后显示“SUCCESS To compile: ……”，此时才能进行后面的步骤，否则长时间的 make 后会报错。

编译 kaldi 的源码：

~/kaldi/src$ make depend

~/kaldi/src$ make

make 的时间较长，大约半个小时到一个小时，如果编译过程中未出现红色的 error，最后出现“Done”，表明编译成功。

以上是 kaldi 工具安装的过程，每步执行都成功，则安装成功，可以利用工具开始语音识别实践的学习。

3.7.3.2 实践语音识别系统

下面我们介绍如何利用 kaldi 工具实践语音识别系统。在 kaldi 工具中的 egs 文件夹中，有许多语音处理相关的实验例子：

- babel : IARPA Babel program 语料库来自巴比塔项目，主要是对低资源语言的语音识别和关键词检索例子，包括普什语， 波斯语，土耳其语，越南语等等。wer 在 50 以上。
- sre08： " Speaker Recognition Evaluations" 说话人识别。
- aurora4：研究鲁棒的语音识别项目。包括说话人分离，音乐分离，噪声分离。
- hkust：香港大学的普通话语音识别。
- callhome_egyptian: 埃及的阿拉伯语语音识别。
- chime_wsj0：chime 挑战项目数据，这个挑战是对电话，会议，远距离麦克风数据进行识别。
- fisher_englist：英语的双声道语音。
- gale_arabic：全球自动语言开发计划中的阿拉伯语。
- gp：global phone 项目，全球电话语音，共有 19 种不同的语言，每种 15-20 小时的语音。
- lre：包括说话人识别，语种识别
- wsj：wall street journal 华尔街日报语料库，是英文的语音。
- swbd：Switchboard 语料库
- tidigits：成年男性，成年女性，孩子说的不同的数字串语音的识别训练。

- voxforge：开源语音收集项目
- timit：不同性别，不同口音的美国英语发音和词汇标注，包括 Texas Instruments (TI) 和 Massachusetts Institute of Technology (MIT)，所以叫 timit
- tedlium: TED 演讲的英语语音数据。
- yesno: 只有 yes、no 两个词的语音识别，是命令词语音识别任务。

这些例子包含了语音处理的多种不同的任务，包括语音识别、说话人识别、带噪语音识别等等。本节实践以 AIShell 数据集训练中文语音识别为例，具体例子可以参照目录 kaldi/egs/aishell/s5 下的 run.sh 脚本，kaldi 将构建语音识别模型的数据准备、特征提取、声学模型训练、语言模型训练、解码、测试都集成到了一个脚本中，下面我们将拆解这些步骤，分别讲解如何一步一步地搭建语音识别系统。

①下载并解压 aishell 178 小时语料库，其中包括音频、对应的转录文本和词典：

```
local/download_and_untar.sh $data $data_url data_aishell || exit 1;
local/download_and_untar.sh $data $data_url resource_aishell || exit 1;
```

②准备词典

```
local/aishell_prepare_dict.sh $data/resource_aishell || exit 1;
```

运行该脚本会生成一个文件夹 data/local/dict 的文件夹，文件夹中包含 lexicon.txt、extra_questions.txt、nonsilence_phones.txt、optional_silence.txt、silence_phones.txt 文件。

- lexicon.txt：文件格式为<word> <phone1> <phone2>的词典，若其中有一个词有不同发音，则会在不同行中出现多次。
- extra_questions.txt：包含那些自动产生的问题集之外的一些问题。你可以看到，所谓一个问题就是一组音素。第一行 sil 是“静音”音素，它被作为发音字典中可选的静音词的表示，即不会出现在某个词中，而是单独成词。
- nonsilence_phones.txt：所有的非静音音素都包含在这个文件中。请注意，nonsilence_phones.txt 中的某些行可能出现一行中有多个音素。这些是同一元音的与重音相关的不同表示。
- optional_silence.txt：文件中只含有一个音素 sil，该音素可在需要的时候出现在词之间。
- silence_phones.txt：包含所有的“静音”音素，在 AIShell 的例子中为一个 sil 音素。

③准备数据。分成 test、dev、train 集

```
local/aishell_data_prep.sh $data/data_aishell/wav $data/data_aishell/transcript || exit 1;
```

在本例中，以数据集推荐的训练集、测试集、验证集来划分。注意，在划分自己的数据集时，要根据说话人来划分句子，训练集、测试集、验证集的说话人不重复。

④词典、语言文件准备，生成对应的数据关系：

```
Phone Sets, questions, L compilation
utils/prepare_lang.sh --position-dependent-phones false data/local/dict \
  "<SPOKEN_NOISE>" data/local/lang data/lang || exit 1;
```

其中，数据关系保存在 data 文件夹里，文件解释如下：

- spk2utt 包含说话人编号和说话人的语音编号的信息，格式为“<speaker-id> <utterance-id>”。
- text 包含语音和语音编号之间的关系，格式为“<utterance-id> <text>”。
- utt2spk 语音编号和说话人编号之间的关系，与 spk2utt 相反，格式为“<utterance-id> <speaker-id>”
- wav.scp 包含了原始语音的路径信息等，格式为“<utterance-id> <wavpath>”

请注意，上述所有文件都需要按照说话人编号进行排序。

⑤提取 MFCC 特征：

```
# Now make MFCC plus pitch features.
# mfccdir should be some place with a largish disk where you
# want to store MFCC features.
mfccdir=mfcc
for x in train dev test; do
  steps/make_mfcc_pitch.sh --cmd "$train_cmd" --nj 10 data/$x exp/make_mfcc/$x
 $mfccdir || exit 1;
  steps/compute_cmvn_stats.sh data/$x exp/make_mfcc/$x $mfccdir || exit 1;
  utils/fix_data_dir.sh data/$x || exit 1;
done
```

提取 MFCC 特征的过程分为两步，先通过 steps/make_mfcc.sh 提取 MFCC 特征，再通过 steps/compute_cmvn_stats.sh 计算倒谱均值和方差归一化。

生成了两个文件夹：mfcc 和 exp/make_mfcc，其中 mfcc 里主要保存了提取的特征，主要是 .ark 和 .scp 文件，其中 .scp 文件里的内容是语音段和特征对应，而真正的特征保存在 .ark 文件里。而 exp/make_mfcc 里保存了日志，即 .log 文件。

在 steps/make_mfcc.sh 里用到的最主要的命令就是 compute-mfcc-feats 和 copy-feats，其在 src 里编译好的。

用下面的命令可以查看提出的 MFCC 特征

```
copy-feats ark:mfcc/raw_mfcc_train.1.ark ark,t:-
```

另外，若要检查特征的维度，可以用 feats-to-dim 命令来查看：

```
feats-to-dim scp: <.scp 文件的路径> -
```

⑥单音素训练：

有了特征，就可以开始训练声学模型了，首先用以下命令进行单音素训练

```
steps/train_mono.sh --cmd "$train_cmd" --nj 10 \
   data/train data/lang exp/mono || exit 1;
```

之后会在 exp 文件夹下产生一个 mono 的目录，其中以 .mdl 结尾的文件保存了模型的参数。使用下面的命令可以查看模型的内容。

```
gmm-copy --binary=false exp/mono/0.mdl - | less
```

有关上下文的信息可以查看 tree 文件：

```
copy-tree --binary=false exp/mono/tree - | less
```

⑦构建单音素解码图：

```
# Monophone decoding
utils/mkgraph.sh data/lang_test exp/mono exp/mono/graph || exit 1;
```

mkgraph.sh 主要生成了 HCLG.fst 和 words.txt 这两个重要的文件，其中 HCLG.fst 保存了完全的 fst，而 words.txt 是词典，它将词的 id 号映射到对应的字符上。后续识别主要利用了三个文件，分别是 final.mdl、HCLG.fst、words.txt。

⑧解码：分别针对开发集和测试集解码

```
steps/decode.sh --cmd "$decode_cmd" --config conf/decode.config --nj 10 \
  exp/mono/graph data/dev exp/mono/decode_dev
steps/decode.sh --cmd "$decode_cmd" --config conf/decode.config --nj 10 \
  exp/mono/graph data/test exp/mono/decode_test
```

解码的日志会保存在 exp/mono/decode_dev/log 和 exp/mono/decode_test/log 里。

⑨ Veterbi 对齐

```
# Get alignments from monophone system.
steps/align_si.sh --cmd "$train_cmd" --nj 10 \
  data/train data/lang exp/mono exp/mono_ali || exit 1;
```

⑩三音素模型训练、解码、对齐

```
# train tri2 [delta+delta-deltas]
steps/train_deltas.sh --cmd "$train_cmd" \
 2500 20000 data/train data/lang exp/tri1_ali exp/tri2 || exit 1;

# decode tri2
utils/mkgraph.sh data/lang_test exp/tri2 exp/tri2/graph
steps/decode.sh --cmd "$decode_cmd" --config conf/decode.config --nj 10 \
  exp/tri2/graph data/dev exp/tri2/decode_dev
steps/decode.sh --cmd "$decode_cmd" --config conf/decode.config --nj 10 \
  exp/tri2/graph data/test exp/tri2/decode_test
```

接下来就和训练单音素一样，进行其他模型的训练解码，生成声学模型和语言模型，保存在 exp 文件夹中中。

训练结束后，可以输入下面的命令来查看结果：

```
# getting results (see RESULTS file)
for x in exp/*/decode_test; do [ -d $x ] && grep W
```

参考文献

[1] L. Rabiner and B. Juang, "Fundamentals of speech recognition," in Prentice Hall signal processing series, 1993.

[2] D. Povey, L. Burget, M. Agarwal, P. Akyazi, K. Feng, A. Ghoshal, O. Glembek, N. Goel, M. Karafi´ at, A. Rastrow, R. Rose, P. Schwarz, and S. Thomas, “The subspace Gaussian mixture model - A structured model for speech recognition,” Computer Speech and Language, vol. 25, pp. 404–439, 2011.

[3] G. E. Hinton, S. Osindero, and Y. Teh, "A Fast Learning Algorithm for Deep Belief Nets," Neural Computation, vol. 18, pp. 1527-1554, 2006.

[4] A.-r. Mohamed, G. E. Dahl, and G. E. Hinton, "Deep Belief Networks for phone recognition," 2009.

[5] G. Dahl, D. Yu, l. Deng, and A. Acero, “Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition,” Audio, Speech, and Language Processing, IEEE Transactions on, vol. 20, pp. 30-42, 02/01, 2012.

[6] W. Zaremba, I. Sutskever, and O. Vinyals, “Recurrent Neural Network Regularization,” ArXiv, vol. abs/1409.2329, 2014.

[7] S. Hochreiter, and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, pp. 1735-1780, 1997.

[8] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” ArXiv, vol. abs/1412.3555, 2014.

[9] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks, 2006.

[10] A. Graves, and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” 31st International Conference on Machine Learning, ICML 2014, vol. 5, pp. 1764-1772, 01/01, 2014.

[11] A. Graves, “Sequence Transduction with Recurrent Neural Networks,” ArXiv, vol. abs/1211.3711, 2012.

[12] C.-C. Chiu, T. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, State-of-the-Art Speech Recognition with Sequence-to-Sequence Models, 2018.

[13] A.A. Markov. "Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga". Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete, 2-ya seriya, tom 15, pp 135-156, 1906.

[14] 李航. 统计学习方法[M]. 清华大学出版社, 2012.

[15] Lawrence R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 77 (2), p. 257–286, February 1989.

[16] 韩纪庆, 张磊, 郑铁然. 高等院校信息与通信工程系列教材 语音信号处理[M]// 高等院校信息与通信工程系列教材, 语音信号处理. 清华大学出版社, 2004.

[17] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. Readings in speech recognition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 267–296, 1990.

[18] Rabiner L , Juang B H . Fundamentals of speech recognition[M]. Tsinghua University Press,

1999.
[19] Welch L R . Hidden Markov Models and the Baum-Welch Algorithm[J]. IEEE Information Theory Society Newsletter, 2003, 53(2):194-211.
[20] Chowdhury A , Koval D . Fundamentals of Probability and Statistics[J]. power distribution system reliability practical methods & applications.
[21] 俞栋, 邓力, 俞凯,等. 解析深度学习:语音识别实践[M]// 解析深度学习：语音识别实践. 电子工业出版社, 2016.
[22] Jurafsky D S , Martin J H . Speech and Language Processing[M]// Speech and language processing :. Prentice Hall, 2010.
[23] Zhao J , Zhang X , Ganapathiraju A , et al. Decision Tree-Based State Tying For Acoustic Modeling[J]. 1999.
[24] Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.-r., et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. 2012. 29(6): p. 82-97.
[25] Rumelhart, D.E., Hinton, G.E., Williams, R.J.J.n., Learning representations by back-propagating errors. 1986. 323(6088): p. 533-536.
[26] 韩力群, 康芊. 《人工神经网络理论、设计及应用》——神经细胞、神经网络和神经系统[J]. 北京工商大学学报:自然科学版, 2005, 23(1):52-52.
[27] Chengalvarayan R , Deng L . HMM-based speech recognition using state-dependent, discriminatively derived transforms on mel-warped DFT features[J]. IEEE Transactions on Speech and Audio Processing, 1997, 5(3):243-256.
[28] 杨宁. 基于多 GPU 并行框架的 DNN 语音识别研究[J]. 微电子学与计算机, 2015(6):6-10.
[29] Canevari C , Badino L , Fadiga L , et al. Cross-corpus and cross-linguistic evaluation of a speaker-dependent DNN-HMM ASR system using EMA data[C]// Workshop on Speech Production for Automatic Speech Recognition. 2013.
[30] 李云红, 梁思程, 贾凯莉,等. 一种改进的 DNN-HMM 的语音识别方法[J]. 应用声学, 2019(3):371-377.
[31] Din G M U , Marnerides A K . Short term power load forecasting using Deep Neural Networks[C]// 2017 International Conference on Computing, Networking and Communications (ICNC). IEEE, 2017.
[32] Li J , Yu D , Huang J T , et al. Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM[C]// Spoken Language Technology Workshop. IEEE, 2012.
[33] Zhao T , Zhao Y , Chen X . Building an ensemble of CD-DNN-HMM acoustic model using random forests of phonetic decision trees[C]// International Symposium on Chinese Spoken Language Processing. IEEE, 2014.
[34] Sugiyama M , Sawai H , Waibel A H . Review of TDNN (time delay neural network) architectures for speech recognition[C]// IEEE International Sympoisum on Circuits & Systems. IEEE, 1991.
[35] Hosung Park, Donghyun Lee, Minkyu Lim, Yoseb Kang, Juneseok Oh and Ji-Hwan Kim，A Fast-Converged Acoustic Modeling for Korean Speech Recognition: A Preliminary Study on Time Delay Neural Network. https://arxiv.org/.
[36] Miao Y , Gowayyed M , Metze F . EESEN: End-to-End Speech Recognition using Deep RNN Models and WFST-based Decoding[J]. 2016.
[37] 朱会峰, 贺勇, 雍坤,等. 基于 DNN 与 RNN 声学模型融和的语音识别研究[C]// 第十三届

全国人机语音通讯学术会议(NCMMSC2015)论文集. 2015.
雷锋网 AI 研习社 https://www.leiphone.com/news/201712/6F577yaQueXAppZG.html.
[38] Graves A, Mohamed A, Hinton G. Speech recognition with deep recurrent neural networks[C]//2013 IEEE international conference on acoustics, speech and signal processing. Ieee, 2013: 6645-6649.
[39] Hochreiter S , Schmidhuber J . Long Short-Term Memory[J]. Neural Computation, 1997, 9(8):1735-1780.
[40] Greff K , Srivastava R K , Jan Koutník, et al. LSTM: A Search Space Odyssey[J]. IEEE Transactions on Neural Networks & Learning Systems, 2016, 28(10):2222-2232.
[41] Irie K , Tuske Z , Alkhouli T , et al. LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modeling in Speech Recognition[C]// Interspeech 2016. 2016.
[42] Cho K , Van Merrienboer B , Gulcehre C , et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation[J]. Computer ence, 2014.
[43] Bengio, Yoshua, et al. "A Neural Probabilistic Language Model." Journal of Machine Learning Research, vol. 3, no. 6, 2003, pp. 1137–1155
[44] Mikolov T , Martin Karafiát, Burget L , et al. Recurrent neural network based language model[C]// Interspeech, Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September. DBLP, 2015.
[45] Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850.
[46] Bahdanau D , Cho K , Bengio Y . Neural Machine Translation by Jointly Learning to Align and Translate[J]. Computer ence, 2014.
[47] Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[J]. 2018.
[48] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Polosukhin, I. Attention Is All You Need. [C] In Advances in Neural Information Processing Systems, 2017.
[49] Elman J L. Finding structure in time[J]. Cognitive science, 1990, 14(2): 179-211.
[50] Sundermeyer M , Ralf Schlüter, Ney H . LSTM Neural Networks for Language Modeling[C]// Interspeech. 2012.
[51] Mikolov T, Sutskever I, Deoras A, et al. Subword language modeling with neural networks[J], 2012, 8: 67.
[52] Hwang K, Sung W. Character-level incremental speech recognition with recurrent neural networks[C]//2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2016: 5335-5339.
[53] Young S J, Russell N H, Thornton J H S. Token passing: a simple conceptual model for connected speech recognition systems[M]. Cambridge: Cambridge University Engineering Department, 1989.Si Y, Wang L, Dang J, et al. A Hierarchical Model for Dialog Act Recognition Considering Acoustic and Lexical Context Information[C]//ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020: 7994-7998.
[54] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.
[55] Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners[J]. OpenAI blog, 2019, 1(8): 9.

[56] Young S J, Russell N H, Thornton J H S. Token passing: a simple conceptual model for connected speech recognition systems[M]. Cambridge: Cambridge University Engineering Department, 1989.
[57] https://www.tensorflow.org/tutorials/seq2seq
[58] Mohri, Mehryar, et al. Speech Recognition with Weighted Finite-State Transducers. 2008, pp. 559–584.
[59] http://htk.eng.cam.ac.uk/
[60] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Vesel, “The Kaldi speech recognition toolkit,” IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, 01/01, 2011.
[61] https://github.com/kaldi-asr/kaldi.git
[62] http://www.kaldi-asr.org/doc/
[63] 陈果果、都家宇、那兴宇、张俊博, Kaldi 语音识别实战, 2020.
[64] S. Watanabe et al., "ESPnet: End-to-End Speech Processing Toolkit," ArXiv, vol. abs/1804.00015, 2018.
[65] https://github.com/espnet/espnet