

Lecture 04: 自动语音识别基础 (ASR Foundation)

概率框架、经典声学模型、解码与评价

王帅

南京大学智能科学与技术学院

2026 年 9 月 18 日



南京大学
NANJING UNIVERSITY



智能科学与技术学院
School of Intelligence Science and Technology

- ① 用一句数学表达说明 ASR 到底在求什么
- ② 解释经典系统中声学模型、发音词典、语言模型和解码器的分工
- ③ 解释 HMM 如何处理时间对齐，以及 GMM 与 DNN 如何建模声学状态
- ④ 正确计算 CER / WER，并据此定位系统错误

本讲核心问题

同一段声音可能对应多种文字，系统如何结合**听起来像什么**与**哪句话更合理**作出选择？

任务定义与声学建模

- ① ASR 的任务、挑战与发展脉络
- ② 概率建模与经典系统结构
- ③ 声学前端：MFCC 与 VAD
- ④ HMM-GMM、DNN-HMM

解码、评价与实践

- ⑤ 词典、语言模型与搜索
- ⑥ WER / CER、编辑距离与错误分析
- ⑦ 工具实践与课程作业

先建立直觉，再看公式；每个数学工具都对应一个具体的识别问题。

耳朵收到的只是声音

/wǒ xiǎng mǎi yì zhī bǐ/

- 声音里没有空格，也没有标点
- “一支笔”和“一只笔”听起来很接近
- 口音、语速和噪声会继续增加不确定性

大脑会利用上下文

- ① “买”后面更可能出现物品
- ② “笔”的常用量词是“支”
- ③ 因而选择：**我想买一支笔**

直觉

ASR 不只是辨认声音，还要在许多候选句子中寻找**整体最合理**的一条。



ASR 负责

- 识别说了哪些字或词
- 常附带时间戳、标点或置信度

ASR 本身不等于

- 说话人识别：谁在说
- 语义理解：说话人想做什么

声音本身会变化

- 同一个人：语速、情绪、疲劳程度不同
- 不同的人：音色、口音、方言不同
- 不同设备：麦克风、距离、混响不同

语言本身也有歧义

- 同音词：“权利”与“权力”
- 连读与吞音：“不知道”常说成“不造”
- 专有名词与新词：人名、地名、课程名

环境会“抢话”

风声、键盘声、多人交谈都会与目标语音叠加。

所以

一个好系统既要对声学变化鲁棒，又要理解语言中的规律。

设置	典型输入	主要难点
孤立词识别	“开灯”“暂停”“下一首”	词表小，但短语音信息少
连续语音识别	新闻、课堂录音、有声书	长时依赖、断句与标点
对话语音识别	客服、会议、访谈	重叠说话、口语化、话轮切换
流式语音识别	输入法、实时字幕、车载指令	低延迟，不能等待整句话结束
多语种/方言识别	中英混说、地区口音	语言切换、数据不均衡

评价系统前先问

它服务于哪一种场景？离线模型与实时模型不能只用一个准确率简单比较。



主线

模板匹配 → 概率建模 → 深度声学模型。

重要节点

2010 年前后，深度神经网络替代 GMM 建模声学状态，HMM 继续负责状态转移与时间对齐。

1877 年：爱迪生发明留声机：首次实现声音的记录与再现。

1952 年：Bell Labs 「Audrey」：单说话人数字（0-9）识别，准确率 90

1962 年：IBM 「Shoebox」：可识别 16 个英语词与数字，支持简单算术指令。

1970 年代：DARPA 语音理解计划：引入 HMM，出现可识别约 1000 词的 Harpy 系统。

1980 年代：统计方法主流化：HMM-GMM 框架确立；IBM Tangora 支持约 20k 词汇。

1996 年: Dragon Dictate 发布: 面向消费级的商用听写产品。

2009 年: 深度学习进入 ASR: DNN-HMM 混合模型显著提升准确率 (Hinton 等)。

HMM-GMM 与 DNN-HMM 的比较



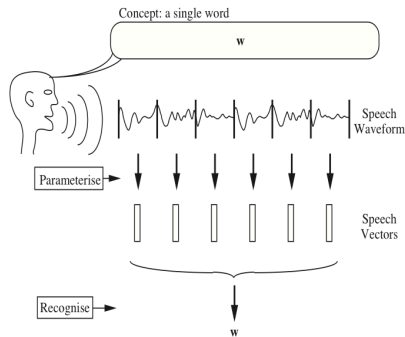
	HMM-GMM	DNN-HMM
声学表示	人工特征	人工特征为主
核心模型	HMM + GMM	HMM + 神经网络
词典	必需	通常必需
语言模型	N-gram	N-gram / 神经 LM
训练方式	分模块	分模块

不变的核心

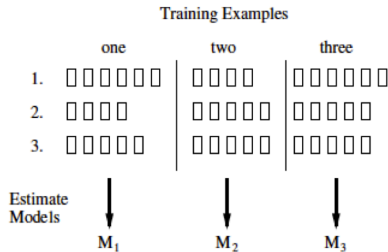
两种系统都保留 HMM、发音词典、语言模型和解码器，主要区别在于声学状态的建模方式。

从孤立词到连续语音识别

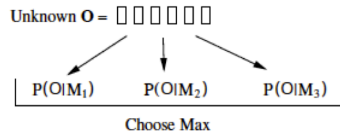
早期思路：为每个候选词建立声学模型



(a) Training



(b) Recognition



设语音经过前端处理后得到特征序列

$$\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_T),$$

候选文字序列为

$$\mathbf{W} = (w_1, w_2, \dots, w_N).$$

ASR 要寻找后验概率最大的文字序列：

$$\mathbf{W}^* = \arg \max_{\mathbf{W}} P(\mathbf{W} \mid \mathbf{O})$$

读法

在已经听到这段声音 $\mathbf{O}$ 的条件下，哪一句文字 $\mathbf{W}$ 最可能？

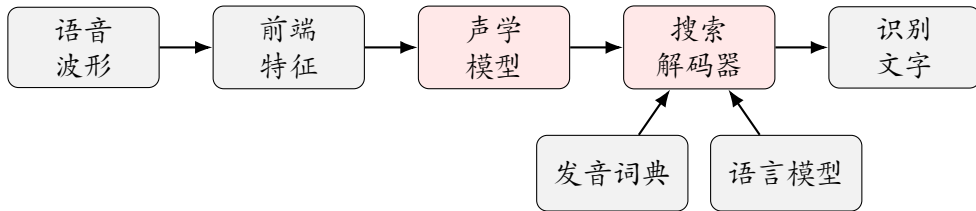
$$\begin{aligned} \mathbf{W}^* &= \arg \max_{\mathbf{W}} P(\mathbf{W} \mid \mathbf{O}) \\ &= \arg \max_{\mathbf{W}} \frac{P(\mathbf{O} \mid \mathbf{W}) P(\mathbf{W})}{P(\mathbf{O})} \\ &= \arg \max_{\mathbf{W}} \underbrace{P(\mathbf{O} \mid \mathbf{W})}_{\text{听起来像不像 这句话通不通顺}} \underbrace{P(\mathbf{W})} \end{aligned}$$

声学模型

给定一段文字，它产生当前声音的可能性有多大？

语言模型

不看声音，这句话在语言中本来有多常见？



- ① 前端把波形变成更紧凑的特征序列
- ② 声学模型估计每个时刻可能对应什么音
- ③ 词典连接“音”和“词”，语言模型提供上下文偏好
- ④ 解码器在海量候选路径中寻找总分最高的一条

经典统计系统：模块如何连接

大词汇连续语音识别系统架构



$$\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} p(\mathbf{O} | \mathbf{W}) P(\mathbf{W})$$

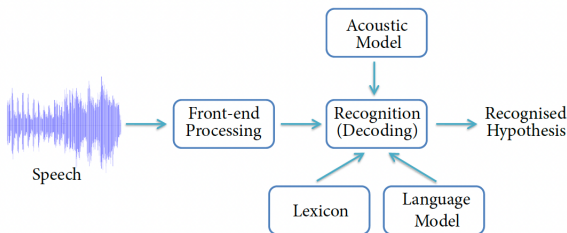
$$\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} p(\mathbf{A} | \mathbf{O}) p(\mathbf{O} | \mathbf{L}) P(\mathbf{L} | \mathbf{W}) P(\mathbf{W})$$

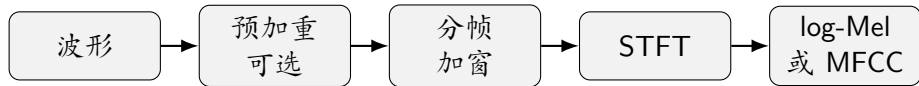
W: 词序列

L: 声学建模单元序列

O: 音频样本序列

A: 声学特征序列



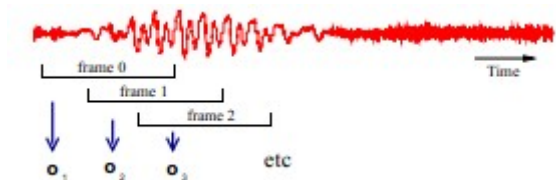


- 典型帧长约 25 ms，帧移约 10 ms
- 特征保留 “什么时候出现了哪些频率成分”
- GMM-HMM 常用 MFCC 及其差分特征；DNN-HMM 也可使用 log-Mel 滤波器组特征

与上一讲的关系

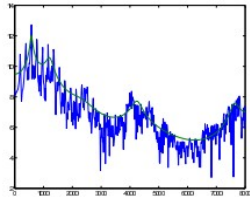
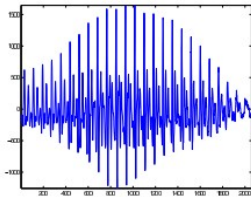
Lecture 03 解决 “怎样表示声音”；本讲从这些特征出发，解决 “怎样得到文字”。

原始语音通过信号处理的方法转换成**特征向量**的序列。



- 特征提取是一个**确定**的过程，即 $p(\mathbf{A}|\mathbf{O}) = \delta(\mathbf{A}, \hat{\mathbf{A}}(\mathbf{O}))$
- 降低信息率，但是保留有用信息
- 去除噪声或者其他的无关信息
 - 识别元音：最低的两个共振峰
 - 识别性别：音调 (pitch) 或者基音周期频率

语音识别中常见的声学特征



我们往往用短时谱信息来描述语音信号。一些有用的短时谱分析技术用于提取有用的特征，包括：

- 线性预测系数 (LPC)
- 滤波器组系数 (Fbank)
- 梅尔频率倒谱系数 (MFCC)
- 等

前端处理：特征提取 $p(A|O)$

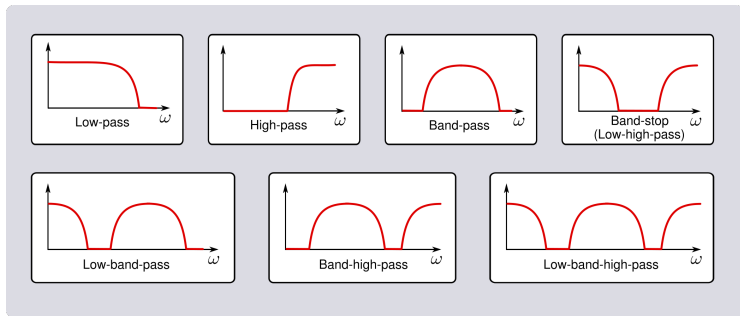
Filter 滤波器



滤波器 一般是一个设备或者处理过程, 它可以从原始信号中将那一些不想要的成分或者特征去除

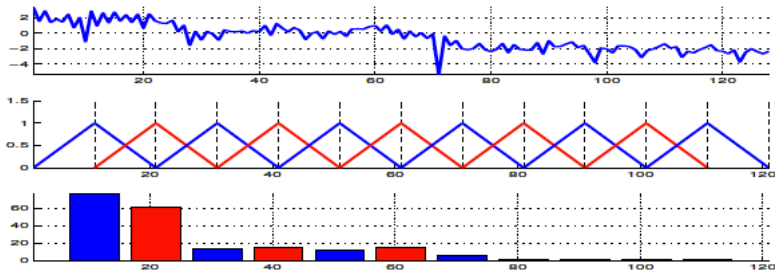
滤波操作一般情况下作用在频率域上:

$$Y(\omega) = H(\omega)X(\omega)$$



前端处理：特征提取 $p(A|O)$

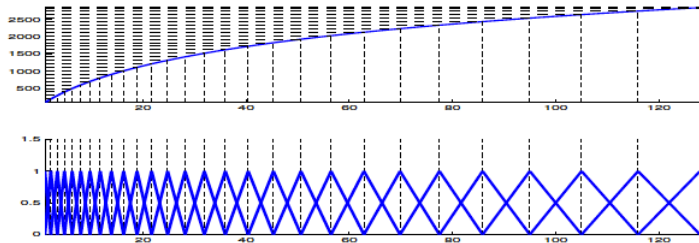
Filter Bank 系数 (FBank)



- 通过短时傅里叶变换得到的幅值谱包含太多的信息
- **Filter bank** 是一系列带通滤波器 (一般用三角窗滤波器)
- 每一个带通滤波器输出一个 FBank 系数, 它等于此带通滤波器内的信号加权和

前端处理：特征提取 $p(A|O)$

梅尔域 (Mel Scale)



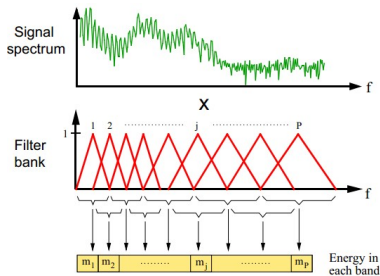
Mel scale 是一个基音感知域，它通过听音者可以区分两种纯音频率的差距来作为标度。Mel 域和线性频域之间通过一个非线性的映射函数可以相互转换：

$$\text{Mel}(f) = 2595 \log_{10} \left(1 + \frac{f}{700} \right)$$

注意：在 Mel 域中，低频部分具有更高的分辨率。

前端处理：特征提取 $p(A|O)$

梅尔域的 FBank 系数



$$m_i = \sum_{k=f_i}^{F_i} s(k) T_i(k)$$

其中 m_i 是第 i^{th} 个 FBank 系数, f_i 和 F_i 分别是三角滤波器的开始和结束频率, $s(k)$ 是在频点 k 的谱的能量 (有时为谱的幅值), $T_i(k)$ 是三角滤波器的值。

前端处理：特征提取 $p(A|O)$

梅尔频率倒谱系数 (MFCC)



MFCC 在许多语音处理的领域被广泛使用。它们从 Mel 域的 FBank 系数衍生得到：

- ① 取对数 **logarithm** 得到 N 个对数域的 FBank 系数
- ② 计算倒谱 **Cepstral** 系数，使用 **Discrete Cosine Transform (DCT)** 变换

$$c_n = \sqrt{\frac{2}{N_{fb}}} \sum_{j=1}^{N_{fb}} \log(m_j) \cos\left(\frac{\pi n}{N_{fb}}(j - 0.5)\right) \quad n = 1, 2, \dots, N_{mfcc}$$

其中 c_n 是第 n^{th} 个 MFCC 系数, m_j 是第 j^{th} 个 mel 域的 FBank 系数, N_{fb} 和 N_{mfcc} 分别是 FBank 系数和最终的 MFCC 系数的数量 (通常情况下 $N_{mfcc} = 12$, N_{fb} 变化范围是 20-30, 甚至 40).

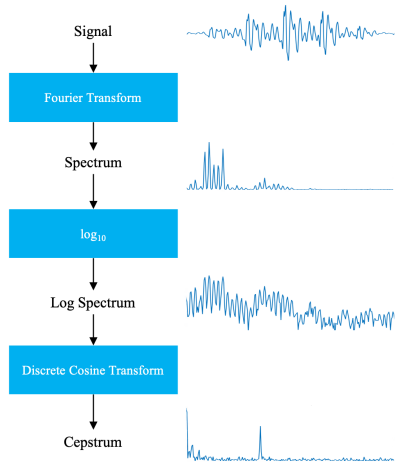
从频谱 (Spectrum) 到倒谱 (Cepstrum)



语音短时谱能量没有直接用来作为特征参数，因为

- 语音的能量谱不符合高斯分布
- 所有系数对响度很敏感
- 各个系数之间都高度相关

Discrete Cosine Transform (DCT) 能够有效地去除不同系数之间的相关性。

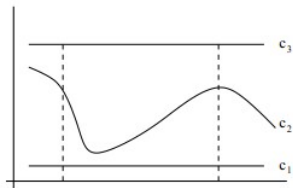


前端处理：特征提取 $p(A|O)$

语音识别中的动态特征



MFCC 特征很好地描述了即时语音信号谱（静态），但是不能描述信号的动态特性。



这个不足可以通过在特征向量中添加静态系数的差分特征来得到改善。

$$\mathbf{o} = \begin{bmatrix} \mathbf{c} \\ \Delta \mathbf{c} \\ \Delta \Delta \mathbf{c} \end{bmatrix}$$

- 简单的差分系数可以计算如下：

$$\Delta_n = \frac{c_{n+\delta} - c_{n-\delta}}{2\delta}$$

- 更鲁棒的估计是通过一系列语音帧的最佳线性回归系数来得到：(这里是 $2\sigma + 1$)

$$\Delta_n = \frac{\sum_{i=1}^{\delta} i(c_{n+i} - c_{n-i})}{2 \sum_{i=1}^{\delta} i^2}$$

- 更高阶的差分系数可以通过以上过程的迭代形式来获得：

$$\Delta\Delta_n = \frac{\sum_{i=1}^{\delta} i(\Delta_{n+i} - \Delta_{n-i})}{2 \sum_{i=1}^{\delta} i^2}$$

- 通常使用一阶与二阶差分，较少使用三阶差分

- ① 预加重 (Pre-emphasis): 提升高频分量。
- ② 分帧 (Framing) 与加窗 (Windowing): 将信号切分为约 25ms 的短帧。
- ③ 离散傅里叶变换 (DFT/FFT): 将每一帧从时域转换到频域, 得到频谱。
- ④ 梅尔滤波器组 (Mel Filter-bank): 对频谱进行滤波, 模拟人耳听觉特性。
- ⑤ 对数运算 (Log): 对滤波器组能量取对数。
- ⑥ 离散余弦变换 (DCT): 解除特征之间的相关性, 得到最终的 MFCC 系数。

最终特征向量

MFCC (12-13 维) + 能量 (1 维) + 一阶差分 (Deltas) + 二阶差分 (Double-Deltas) $\rightarrow$ 39 维特征向量

为什么需要 VAD

- 去掉长时间静音，减少计算
- 确定一句话的开始与结束
- 避免把噪声误识别成文字

两类做法

- 简单方法：能量、过零率 + 阈值
- 学习方法：逐帧判断 speech / non-speech

阈值法的两难

阈值太低：键盘声也被当成语音。

阈值太高：轻声、句尾容易被截断。

实践提示

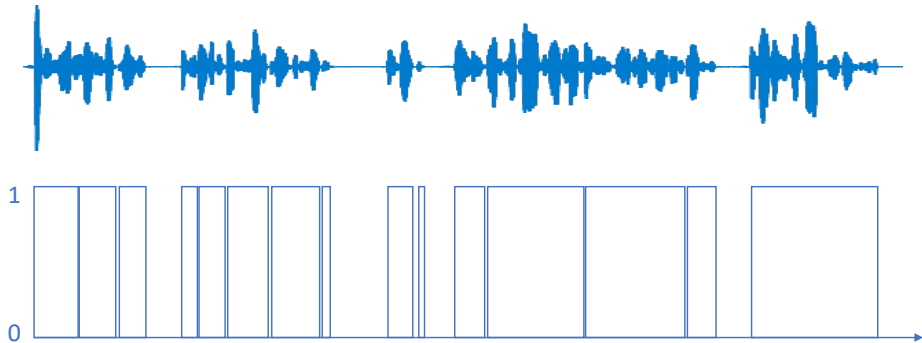
VAD 看似只是前处理，却会直接决定后面 ASR 能看到什么。

前端处理：语音端点检测 (VAD)

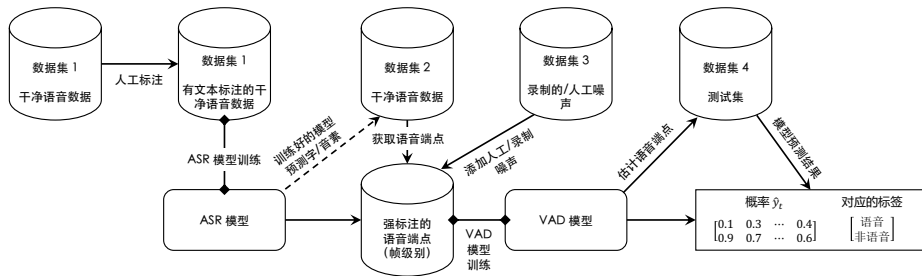


语音端点检测 — Voice Activity Detection (VAD)

- 主要任务：检测一段音频中是否存在人类的语音
- 作用：
 - 减少不必要的语音处理，节省运算开销和功耗
 - 也能提升后续语音处理的性能（减少了不必要的非语音噪声输入）



传统的有监督 VAD 模型训练流程



前端处理：语音端点检测 (VAD)

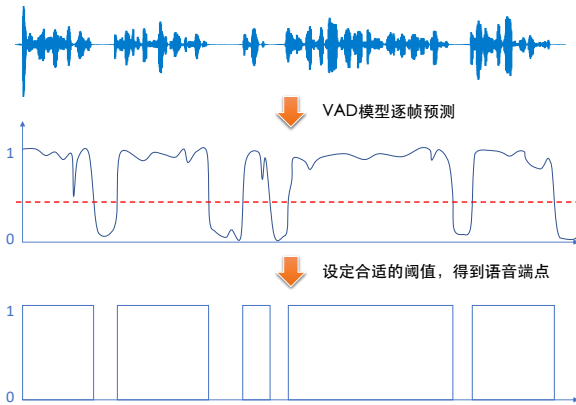
传统的有监督 VAD 模型训练流程



- 1. 采集干净的单人语音数据，尤其是音频质量相差不大的数据。确保数据中仅包含固定种类的一种/多种语言。
- 2. 手动对这些数据进行文本标注。
- 3. 在这些数据上训练一个 ASR 模型。
- 4. 用训练好的 ASR 模型来预测新的（通常干净）数据集中的音频是否存在语音，以得到帧级别的强标注。
 - ASR 模型可以提供音素级别的对齐，进而转换为二值的语音指示（有语音的帧设为 1，静音段设为 0）
- 5. 在新数据集上用前一步得到的强标注训练一个 VAD 模型（DNN/CNN/RNN 等）。

前端处理：语音端点检测 (VAD)

VAD 模型测试阶段



输入

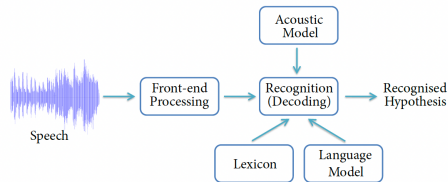
一串声学特征 $\mathbf{o}_1, \dots, \mathbf{o}_T$

输出

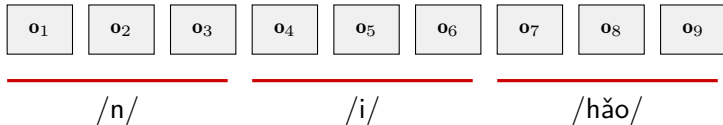
每一帧属于各个声学状态或音素的分数

例: /mā/

模型需要依次发现 /m/ 的闭唇鼻音特征和 /a/ 的元音共振峰, 同时容忍说话速度变化。



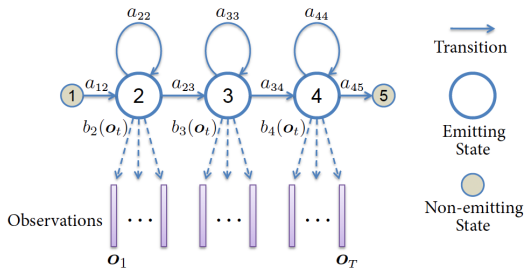
声音到文字之间存在多层对应关系



- 语音有很多帧，但音素或字的数量少得多
- 同一个音素可以持续不同帧数
- 训练时往往只有整句文字，没有每一帧的准确标签

这就是对齐问题

模型必须一边判断“是什么音”，一边判断“这个音持续到哪里”。



- 一个音素常拆成“开始—稳定—结束”三个状态
- 自环表示停留：同一个音可以说得更久
- 向前跳转表示发音过程按时间推进

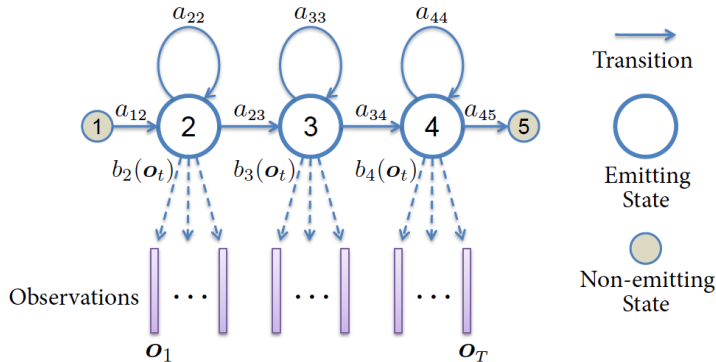
HMM 擅长什么

它把**内容**与**持续时间**拆开，让长短不同的语音都能与同一音素序列对齐。

为什么 HMM 适合语音？



- 语音信号具有时序性和序列性，其特性随时间演进。
- HMM 能够优雅地对这种时变序列进行建模。
- 语音的观测值 (声学特征) 是连续的，但其背后的本质状态是离散的 (如音素)，这与 HMM 的假设一致。



一个 HMM 模型 λ 由五个要素定义:

- 状态集合 $S = \{s_1, \dots, s_N\}$: 模型的内部状态, 如音素的不同阶段。
- 观测序列 $X = (\mathbf{x}_1, \dots, \mathbf{x}_T)$: 每一时刻的声学特征向量。
- 转移概率矩阵 $A = [a_{ij}]$, 其中 $a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$ 。
- 发射概率 (观测概率) $B = \{b_j(\mathbf{x}_t)\}$, 其中 $b_j(\mathbf{x}_t) = P(\mathbf{x}_t | q_t = s_j)$ 。
- 初始状态分布 $\pi = \{\pi_i\}$, 其中 $\pi_i = P(q_1 = s_i)$ 。

$$\lambda = (A, B, \pi) \quad (1)$$

- ① 评估 (Evaluation): 给定模型 λ , 计算观测序列 X 出现的概率 $P(X|\lambda)$.
 - 算法: 前向算法 (Forward Algorithm)。
- ② 解码 (Decoding): 给定模型 λ 和观测序列 X , 找到最可能的状态序列 Q .
 - 算法: 维特比算法 (Viterbi Algorithm)。
- ③ 学习 (Learning): 给定观测序列 X , 调整模型参数 λ 使得 $P(X|\lambda)$ 最大。
 - 算法: Baum-Welch 算法 (即 EM 算法在 HMM 中的特例)。

目标: 计算 $P(X|\lambda)$

定义前向变量 $\alpha_t(i) = P(\mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i | \lambda)$, 表示到时间 t 观测到序列 $\mathbf{x}_1, \dots, \mathbf{x}_t$ 且处于状态 s_i 的概率。

1. 初始化:

$$\alpha_1(i) = \pi_i b_i(\mathbf{x}_1) \quad (2)$$

2. 递推:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(\mathbf{x}_{t+1}) \quad (3)$$

3. 终止:

$$P(X|\lambda) = \sum_{i=1}^N \alpha_T(i) \quad (4)$$

目标：找到最优状态序列 Q^*

定义 $\delta_t(i) = \max_{q_1, \dots, q_{t-1}} P(q_1, \dots, q_t = s_i, \mathbf{x}_1, \dots, \mathbf{x}_t | \lambda)$, 表示到时间 t 且处于状态 s_i 的所有路径中的最大概率。

1. 初始化:

$$\delta_1(i) = \pi_i b_i(\mathbf{x}_1), \quad \psi_1(i) = 0 \quad (5)$$

2. 递推:

$$\delta_t(j) = \max_i [\delta_{t-1}(i) a_{ij}] b_j(\mathbf{x}_t) \quad (6)$$

$$\psi_t(j) = \arg \max_i [\delta_{t-1}(i) a_{ij}] \quad (\text{记录路径}) \quad (7)$$

3. 终止与回溯: 从 $\arg \max_i \delta_T(i)$ 开始, 利用 $\psi_t(j)$ 回溯, 得到最优状态序列。

E-Step: 计算期望

利用前向变量 $\alpha_t(i)$ 和后向变量 $\beta_t(i)$ 计算期望值。

- 在时刻 t 处于状态 s_i 的概率:

$$\gamma_t(i) = P(q_t = s_i | X, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(X|\lambda)} \quad (8)$$

- 在时刻 t 处于状态 s_i 且在 $t+1$ 转移到 s_j 的概率:

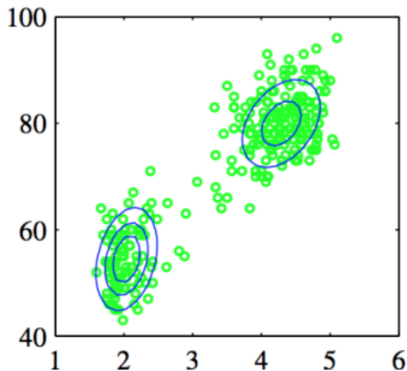
$$\xi_t(i, j) = P(q_t = s_i, q_{t+1} = s_j | X, \lambda) = \frac{\alpha_t(i)a_{ij}b_j(\mathbf{x}_{t+1})\beta_{t+1}(j)}{P(X|\lambda)} \quad (9)$$

M-Step: 最大化期望, 重估参数

$$\pi_i^{\text{new}} = \gamma_1(i) \quad (10)$$

$$a_{ij}^{\text{new}} = \frac{\sum_t \xi_t(i, j)}{\sum_t \gamma_t(i)} \quad (11)$$

$$b_j^{\text{new}}(\cdot) \leftarrow \text{根据 } \sum_t \gamma_t(j) \text{ 和 } \mathbf{x}_t \text{ 更新} \quad (12)$$



- 同一音素由不同人说，特征分布可能有多个峰
- GMM 用若干高斯分布的加权和描述这些变化

$$p(\mathbf{o} | s) = \sum_{m=1}^M c_m \mathcal{N}(\mathbf{o}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$$

局限

为每个状态手工假设分布形状，面对复杂高维语音时表达能力有限。

为什么用 GMM 作为发射概率模型?



- HMM 中的发射概率 $b_j(\mathbf{x}_t) = P(\mathbf{x}_t | q_t = s_j)$ 需要对声学特征 $\mathbf{x}_t$ 的分布进行建模。
- 声学特征向量 $\mathbf{x}_t$ 是连续的、高维的。
- 单一高斯分布的建模能力太有限，无法捕捉复杂的数据分布。
- 高斯混合模型 (GMM) 是一个万能逼近器 (universal approximator)，理论上可以拟合任意形状连续概率分布。

GMM 公式

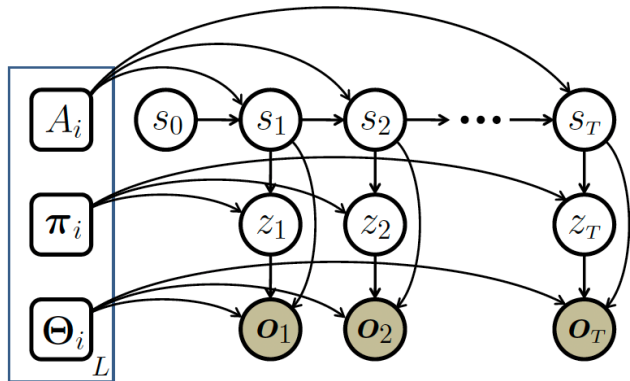
状态 j 的发射概率由 M 个高斯分量混合而成：

$$b_j(\mathbf{x}) = \sum_{k=1}^M w_{jk} \mathcal{N}(\mathbf{x} | \mu_{jk}, \Sigma_{jk}), \quad \text{且} \quad \sum_{k=1}^M w_{jk} = 1 \quad (13)$$

其中, $\mathcal{N}(\mathbf{x} | \mu, \Sigma)$ 是标准的多维高斯分布：

$$\mathcal{N}(\mathbf{x} | \mu, \Sigma) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu) \right] \quad (14)$$

GMM-HMM ASR 系统的整体架构



将 GMM 嵌入 HMM 的 Baum-Welch 算法中, M-Step 变为更新 GMM 的参数。

E-Step: 计算后验概率

计算观测 $\mathbf{x}_t$ 属于状态 j 的第 k 个高斯分量的后验概率:

$$\gamma_t(j, k) = P(q_t = s_j, k_t = k | X, \lambda) \quad (15)$$

M-Step: 更新 GMM 参数

利用加权的观测数据，更新每个高斯分量的权重、均值和协方差：

$$w_{jk}^{\text{new}} = \frac{\sum_t \gamma_t(j, k)}{\sum_t \sum_m \gamma_t(j, m)} \quad (16)$$

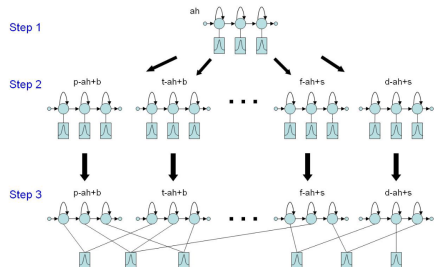
$$\mu_{jk}^{\text{new}} = \frac{\sum_t \gamma_t(j, k) \mathbf{x}_t}{\sum_t \gamma_t(j, k)} \quad (17)$$

$$\Sigma_{jk}^{\text{new}} = \frac{\sum_t \gamma_t(j, k) (\mathbf{x}_t - \mu_{jk}^{\text{new}})(\mathbf{x}_t - \mu_{jk}^{\text{new}})^\top}{\sum_t \gamma_t(j, k)} \quad (18)$$

- /a/ 在不同前后音素之间，发音会略有不同
- 三音素把“左邻—当前—右邻”一起建模
- 组合数量会爆炸，需要决策树共享相似状态

例

/k-a+t/ 表示：当前音素 /a/，左邻是 /k/，右邻是 /t/。



数据不足的状态通过“聚类”共享参数

为什么需要上下文相关模型？

同一个音素在不同上下文中的发音（声学实现）差异巨大，这种现象称为协同发音 (Coarticulation)。

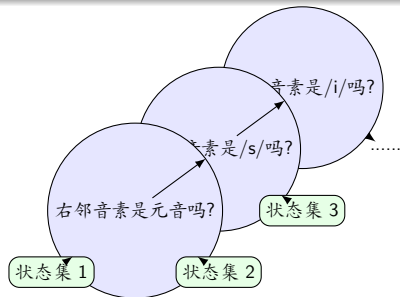
- 例如，/i/ 在 *see* (/s i/) 和 *she* (/sh i/) 中的频谱完全不同。
- 解决方案：使用三音素 (Triphone)，如 s-i+y，同时考虑左、右相邻的音素。

问题：参数爆炸

如果英语有 40 个音素，理论上三音素的数量将达到 $40^3 = 64,000$ 。大多数三音素在训练数据中从未出现过。

核心思想

将声学特性相似的三音素状态“绑定”在一起，共享 GMM 参数。



- 决策树问题: 基于音素的语音学属性提问 (如: 是元音吗? 是塞音吗?)。
- 分裂准则: 选择能带来最大似然增益 (Likelihood Gain) 的问题来分裂节点。

最大似然 (ML) 训练的缺陷

ML 训练 ($\arg \max P(X|C)$) 的目标是让正确文本的似然度最大化, 但它完全忽略了其他错误的、有竞争力的文本。

判别式训练的目标

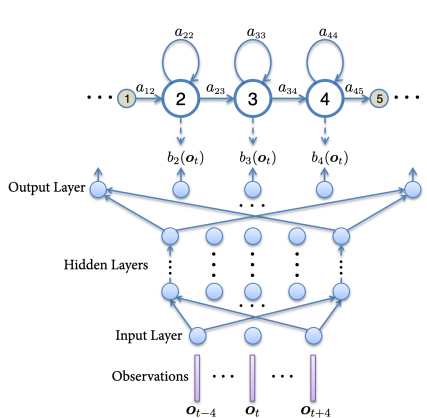
直接优化与最终评估指标 (如词错误率) 更相关的目标。

- 最大互信息 (MMI): 最大化正确文本的后验概率。

$$\mathcal{F}_{\text{MMI}}(\lambda) = \sum_r \log \frac{P_\lambda(X_r|C_r)P(C_r)}{\sum_{C'} P_\lambda(X_r|C')P(C')} \quad (19)$$

- 最小音素/词错误率 (MPE/MWE): 直接以降低音素或词错误为目标。

- 脆弱的“流水线”式架构：每一模块的错误都会累积和传播。
- 依赖大量人工设计的启发式规则：如决策树的问题集、HMM 拓扑结构等。
- GMM 的建模能力不足：难以有效捕捉声学特征在高维空间中的非线性流形结构。
- 训练目标不一致：各模块分别构建或训练，局部优化目标与最终的识别错误率可能不一致。



- HMM 继续负责状态跳转与时间对齐
- 神经网络取代 GMM，估计声学状态概率
- 多层非线性变换能学习更复杂的判别边界

一句话概括

HMM 管“先后与多久”，DNN 管“这一帧像什么”。

中文示例

南京	nán jīng
银行	yín háng
行走	xíng zǒu

英文示例

speech	S P IY CH
data	D EY T AH / D AE T AH

- 一个词可能有多个读音
- 一个读音也可能对应多个词
- 未登录词需要借助字词切分或发音生成

词典的角色

它不是“查意思”，而是规定文字候选如何展开成声学模型能够评分的音素序列。

对句子 $\mathbf{W} = (w_1, \dots, w_N)$, 链式法则给出

$$P(\mathbf{W}) = \prod_{i=1}^N P(w_i \mid w_1, \dots, w_{i-1}).$$

声学上接近

“今天的天气很好”

“今天的天气很好”

语言模型偏好

“天气”在语料中更常见，也与“很好”更搭配。

作用

语言模型用上下文消除同音歧义，并让输出更符合真实语言习惯。

三元语言模型作近似：

$$P(w_i \mid w_1, \dots, w_{i-1}) \approx P(w_i \mid w_{i-2}, w_{i-1}).$$

例：“南京大学位于_____”

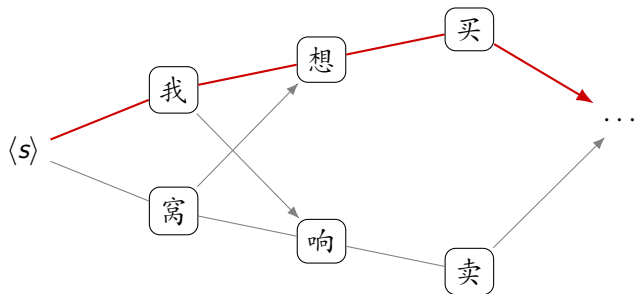
若训练语料中“大学位于南京”经常出现，模型会给“南京”较高概率。

优点

统计、存储和解码都比较直接。

局限

看不到更远的上下文，没见过的组合还会出现零概率。



- 穷举所有句子不可能：候选数量随长度指数增长
- Viterbi / beam search 逐步扩展候选，及时剪掉明显较差的路径
- 每条路径同时累积声学分数和语言分数

实际解码常在对数域中计算：

$$\text{score}(\mathbf{W}) = \log P(\mathbf{O} \mid \mathbf{W}) + \lambda \log P(\mathbf{W}) + \beta |\mathbf{W}|.$$

- λ ：语言模型权重。过大时输出很流畅，却可能“没听声音”
- β ：插词惩罚或奖励。用于调节输出偏长还是偏短
- 参数通常在开发集上选择，不能在测试集上反复调

工程直觉

声学模型、语言模型的数值尺度不同，解码器必须让二者“音量合适”才能合作。

英文常用 WER

reference: we like speech
hypothesis: we love speech

以**词**为单位对齐

中文常用 CER

参考：语音识别
识别：语音识**别**

通常以**汉字**为单位对齐

公平比较的前提

大小写、标点、数字写法和中英文切分规则必须统一，这一步叫文本规范化。

类型	参考文本	识别结果	记号
替换	我喜欢 语音	我喜欢 语言	$S = 1$
删除	我 很 喜欢语音	我喜欢语音	$D = 1$
插入	我喜欢语音	我 很 喜欢语音	$I = 1$

为什么需要对齐

识别结果一旦漏掉一个词，后面的词不能简单按位置逐个比较；我们要寻找总编辑次数最少的对齐方式。

令 $d(i, j)$ 表示参考序列前 i 个单元与识别序列前 j 个单元的最小编辑次数：

$$d(i, j) = \min \begin{cases} d(i-1, j) + 1, & \text{删除,} \\ d(i, j-1) + 1, & \text{插入,} \\ d(i-1, j-1) + \mathbb{I}[r_i \neq h_j], & \text{匹配或替换.} \end{cases}$$

边界条件： $d(i, 0) = i$, $d(0, j) = j$ 。

直觉

到达表格中的每个格子只有三种最后一步：删一个、插一个，或让两个单元对齐。

```
int LevenshteinDistance(char s[], char t[])
{
    int m = s.length, n = t.length;
    int d[][] = new int[m+1][n+1];
    for (int i=0; i<=m; i++) d[i][0] = i;
    for (int j=0; j<=n; j++) d[0][j] = j;
    for (int i=0; i<m; i++) {
        for (int j=0; j<n; j++) {
            int cost = (s[i] == t[j]) ? 0 : 1;
            d[i+1][j+1] = minimum (d[i-1][j]    + 1,           // deletion
                                   d[i][j-1]    + 1,           // insertion
                                   d[i-1][j-1] + cost);          // substitution
        }
    }
    return d[m][n];
}
```

Levenshtein Distance 示例

SATURDAY v.s. SUNDAY



		S	A	T	U	R	D	A	Y
	0	1	2	3	4	5	6	7	8
S	1								
U	2								
N	3								
D	4								
A	5								
Y	6								

- 初始化: 填满第一行和第一列
 - 第一行: 插入 (insertion) 错误的数量
 - 第一列: 删除 (deletion) 错误的数量
- 参考标注: SUNDAY
- 假设标注: SATURDAY

Levenshtein Distance 示例

SATURDAY v.s. SUNDAY



		S	A	T	U	R	D	A	Y
	0	1	2	3	4	5	6	7	8
S	1	0	1						
U	2								
N	3								
D	4								
A	5								
Y	6								

- 考虑 entry ($i = 1, j = 1$)

- 插入: $d[1][0] + 1 = 2$
- 删除: $d[0][1] + 1 = 2$
- 替代: $d[0][0] + 0 = 0$

- 最小代价: $\text{cost} = 0$

- 考虑 entry ($i = 1, j = 2$)

- 插入: $d[1][1] + 1 = 1$
- 删除: $d[0][2] + 1 = 3$
- 替代: $d[0][1] + 1 = 2$

- 最小代价: $\text{cost} = 1$

Levenshtein Distance 示例

SATURDAY v.s. SUNDAY



		S	A	T	U	R	D	A	Y	
		0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7	
U	2	1	1	2	2	3	4	5	6	
N	3	2	2	2	3	3	4	5	6	
D	4	3	3	3	3	4	3	4	5	
A	5	4	3	4	4	4	4	3	4	
Y	6	5	4	4	5	5	5	4	3	

因此, 最优的 *Levenshtein distance* 是 3 (2 insertions and 1 substitution)

REF	S			U	N	D	A	Y
HYP	S	A	T	U	R	D	A	Y
Edits		I	I		S			

参考序列长度为 N ，最优对齐中有替换 S 、删除 D 、插入 I ：

$$\text{WER} = \frac{S + D + I}{N} \times 100\%$$

中文按字计算时，同样的公式称为 CER。

例

参考：we like speech recognition ($N = 4$)

识别：we love recognition today

一种最优对齐为： $S = 1$ (like $\rightarrow$ love), $D = 1$ (speech), $I = 1$ (today)

$$\text{WER} = \frac{1 + 1 + 1}{4} = 75\%.$$

WER 可大于 100% 的情形



设参考文本只有两个词：

thank you

系统却输出：

thanks to all of you

最优对齐可能包含 1 次替换和 3 次插入：

$$\text{WER} = \frac{1 + 0 + 3}{2} = 200\%.$$

不要把 WER 当成准确率

WER 是“每个参考词平均产生多少编辑错误”，插入错误没有上限，所以 WER 也没有 100% 的上限。

建议分组统计

- 安静 / 噪声 / 远场
- 普通话 / 方言 / 口音
- 短句 / 长句
- 常用词 / 专有名词

错误类型对应线索

- D 多: VAD 截断或模型漏听
- I 多: 噪声触发或插词奖励过强
- 同音 S 多: 语言模型或上下文不足

评价的最终目的

指标不是为了“报一个分数”，而是为了回答：系统错在哪里，下一步该改什么？

理解经典系统

Kaldi

组件完整、控制细，适合学习 DNN-HMM；
上手成本较高。

识别结果评分

jiwer / torchmetrics

计算 WER / CER，用于交叉核对自己实现的评分结果。

工具不会替你定义实验

在比较模型前，先固定数据划分、文本规范化、切分单位和解码参数。

核心功能

- 1 实现 Levenshtein 动态规划
- 2 输出 S, D, I, N 与 WER / CER
- 3 支持中英文与文本规范化
- 4 对多条样本进行批量或并行评分

必须验证

- 手工构造替换、删除、插入用例
- 与一个现有评分库交叉核对
- 在一组经典 ASR 系统的识别输出上运行

注意

报告中要写清中文切分、英文大小写、标点和数字的处理规则。

测试数据

- 英文 500 条：LibriSpeech test set
- 中文 500 条：AISHELL-1
- 具体数据子集与下载方式另行通知

提交内容

- ① 可运行源代码与使用说明
- ② 实验报告：算法、规范化、正确性验证、结果与错误分析
- ③ 文件名：姓名 _ 学号 _hw1.zip

最低自测

空串、完全一致、全插入、全删除、中文标点、英文大小写都应有测试用例。

- ① ASR 的目标是寻找 $\mathbf{W}^* = \arg \max_{\mathbf{W}} P(\mathbf{W} | \mathbf{O})$
- ② 经典系统用声学模型、词典和语言模型共同给候选文字打分
- ③ HMM 处理时间对齐, GMM / DNN 判断声学状态
- ④ 解码器融合声学分数与语言分数, 并通过搜索找到最优候选
- ⑤ CER / WER 来自最小编辑距离; 分数之后还要做分场景错误分析

带走的一句话

语音识别不是“把声音直接翻译成字”, 而是在不确定性中融合多种证据, 寻找最合理的文字路径。

推荐阅读

- Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition*
- Daniel Jurafsky and James H. Martin, *Speech and Language Processing*, ASR 相关章节

致谢

本节课件参考了经典语音识别教材、开源工具文档及相关课程资料。部分内容与图示参考上海交通大学俞凯教授的课程 Slides，在此谨致谢意。

问题与讨论