第五讲：自动语音识别

传统模型：GMM/DNN - HMM 系统

王帅

2025 年 9 月 22 日

南京大學 NANJING UNIVERSITY
智能科学与技术学院 School of Intelligence Science and Technology

1. 引言与背景（回顾）
2. 隐马尔可夫模型
   - HMM 问题一：评估 (Evaluation)
   - HMM 问题二：解码 (Decoding)
   - HMM 问题三：学习 (Learning)
3. GMM-HMM
4. 从单音素到上下文相关模型
5. DNN-HMM 系统
6. 总结与展望

引言：让机器听懂世界

南京大学 NANJING UNIVERSITY 智能科学与技术学院 School of Intelligence Science and Technology

什么是语音识别

语音识别 (Automatic Speech Recognition, ASR) 是将人类语音信号转换为可编辑文本的技术。简单来说，就是赋予机器"耳朵"来听，"大脑"来理解的能力。

语音 → 文本

应用场景

- 智能家居
- 语音输入法
- 车载导航
- 在线会议

核心挑战

口音与方言
不同地区的口音和方言差异巨大。

背景噪音
嘈杂环境下语音信号易被干扰。

语速变化
人们说话快慢不一，常有停顿、重复。

口语化表达
日常交流充满省略、倒装等非规范语言。

"如何让模型在这些复杂多变的情况下依然保持高准确率，是语音识别领域持续研究的重点。"

核心原理：语音识别的"三驾马车"

语音识别系统并非单一的黑箱，而是由三个精密协作的组件构成。我们可以将它们比喻为语音识别的"三驾马车"：声学模型、语言模型和解码器。它们各司其职，又紧密配合，共同将人类的语音转化为可理解的文本。

声学模型 (AM)

功能："耳朵"

- 将声学信号转化为音素的概率分布。
- 分析语音的声学特征，如音高、音强。
- 从 MFCC 等特征计算音素或状态概率。

技术演进：GMM → RNN/LSTM → Transformer

语言模型 (LM)

功能："大脑"

- 评估一个词序列出现的概率是否合理。
- 解决同音异义词或发音相似词的歧义。
- 通过上下文判断哪个词序列更通顺。

技术演进：N-gram → RNN → Transformer

解码器 (Decoder)

功能："决策者"

- 结合声学模型和语言模型的输出。
- 从所有可能路径中搜索最优词序列。
- 常用算法：维特比和 Beam Search。

目标：在计算效率和准确率间取得平衡。

声学模型：系统的"耳朵"

什么是声学模型

声学模型 (Acoustic Model, AM) 是语音识别系统中的“耳朵”，将输入的声学信号转化为更高级的、与语言相关的表示，通常是音素 (phoneme) 的概率分布。

声波 → 音素概率

从 GMM 到深度学习的演进

GMM 模型
早期模型，需人工设计特征，表达能力有限。

DNN 模型
自动学习复杂、抽象的声学特征，对人工设计特征依赖有限。

RNN/LSTM
捕捉语音时序依赖关系，适合处理长序列。

Transformer
并行处理特征，捕捉全局依赖关系。

“深度学习模型能够自动学习更复杂、更抽象的声学特征，从而显著提高了识别的准确性。”

1. 概率计算
   计算声学特征属于某个特定音素或 HMM 状态的概率。
2. 示例
   以“你好”为例，模型会分析“你”和“好”的声学特征，判断它们最可能对应的音素。

贝叶斯定理 (Bayes' Rule)

$$P(\mathbf{W} \mid \mathbf{O}) = \frac{p(\mathbf{O} \mid \mathbf{W})P(\mathbf{W})}{p(\mathbf{O})} = \frac{p(\mathbf{O} \mid \mathbf{L})\, P(\mathbf{L} \mid \mathbf{W})\, P(\mathbf{W})}{p(\mathbf{O})}$$

最大后验概率 (MAP) 决策

由于 $p(\mathbf{O})$ 与 $\mathbf{W}$ 无关，优化目标可写为：

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} \; p(\mathbf{O} \mid \mathbf{L})\, P(\mathbf{L} \mid \mathbf{W})\, P(\mathbf{W})$$

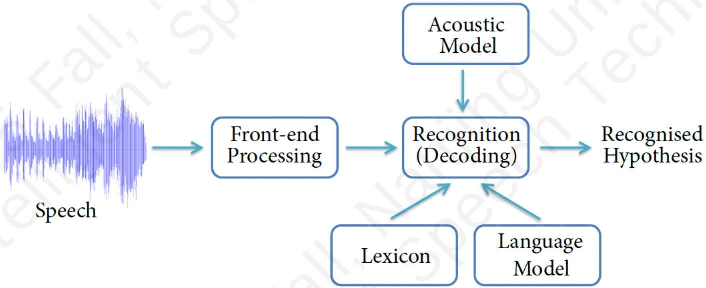
- $p(\mathbf{O} \mid \mathbf{L})$：声学模型（Acoustic Model, AM），声学特征在给定声学建模单元序列下的条件似然。
- $P(\mathbf{L} \mid \mathbf{W})$：字典模型（Lexicon Model），词序列到声学建模单元的映射。
- $P(\mathbf{W})$：语言模型（Language Model, LM），词序列的先验概率（语法与语义合理性）。

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

W: 词序列

L: 声学建模单元序列

O: 音频样本序列



声学模型是一个概率模型，它可以刻画用来描述不同声音的声学特性。

- 语音识别最关键的技术之一
- 概率模型 $p(\mathbf{O}|\mathbf{L})$ 用于刻画不同语音单元，如音素，音节，字，词
- Hidden Markov Model (HMM) 隐马尔科夫模型由于其概念的简单以及数学的完备，被最广泛地采用。

HMM 可以认为是一个最基本的有限状态转录机（FST），它可以将一个用于表示语音的特征向量序列，通过有限状态机，转换成状态机的状态序列（表示音素、音节或词）。

语言模型是一个 概率模型 probabilistic model:

1. 引导搜索算法 (在给定历史的情况下预测下一个词)。
2. 消除声学单元之间的混淆性, 特别是那些声学层相似的单元。
   `Great wine v.s. Grey twine`

语言模型将概率分配到一串要识别的 tokens (通常是词) 上:

- 上下文自由语法:
  `( <s> < one | two | three > </s> )`
- 统计语言模型: n-gram 语言模型

$$P(w_1, w_2, \cdots, w_N)$$

n-gram 统计语言模型和 HMM 声学模型由于容易结合而被广泛地用于语音识别中。

字典模型 $P(\mathbf{L}|\mathbf{W})$

字典模型为声学模型和语言模型之间构建了桥梁。

- 它在词和声学单元之间定义了一个映射。
- 它可以是一个确定化的模型 (**deterministic**)

Word	Pronunciation
TOMATO	t ah m aa t ow t ah m ey t ow
COVERAGE	k ah v er ah jh k ah v r ah jh

- 它也可以是一个概率模型 (**probabilistic**)

Word	Pronunciation	Probability
TOMATO	t ah m aa t ow t ah m ey t ow	0.45 0.55
COVERAGE	k ah v er ah jh k ah v r ah jh	0.65 0.35

- 特征提取 **Feature extraction :** 从原始波形提取声学特征。
- 声学模型 **Acoustic model** $p(\mathbf{O}|\mathbf{L})$: 在给定声学单元（如音素）的条件下对特征分布进行建模。
- 字典 **Lexicon** $P(\mathbf{L}|\mathbf{W})$: 声学建模单元和词之间的映射。
- 语言模型 **Language model** $P(\mathbf{W})$: 产生词序列的概率。
- 解码算法 **Decoding algorithm:** 基于以上各种信息源，找到“最优”词序列。
- 结果评估 **Evaluation:** 对比 *hypotheses*（识别假设标注）和 *reference*（参考标注）

如何对“序列”建模？

回顾我们的问题

- 在上一节我们看到，声学模型 $p(\mathbf{O}|\mathbf{L})$ 的目标是计算给定一个声学建模单元序列 $\mathbf{L}$ 时，观测到声学特征序列 $\mathbf{O}$ 的概率。
- 这里的核心挑战是：语音信号是一个时间序列。它的特性是随时间动态变化的。
- 我们需要一个强大的数学工具来描述这种随时间演进的、具有“记忆”的序列过程。
- 让我们从一个最简单的序列模型开始：马尔可夫链。

定义 (马尔可夫性质)

一个系统在未来某个时刻 $t+1$ 的状态，只取决于它在当前时刻 t 的状态，而与它在 t 时刻之前的任何状态都无关。

$$P(q_{t+1}|q_t, q_{t-1}, \dots, q_1) = P(q_{t+1}|q_t)$$

例子：一个简单的天气模型

- 假设一个地方的天气只有三种状态：{晴天, 阴天, 雨天}。
- 我们通过长期观察，得到了天气变化的规律：
  - 如果今天是晴天，明天有 70% 的概率还是晴天，30% 的概率会阴天。
  - 如果今天是阴天，明天有 40% 的概率会放晴，40% 的概率继续下雨，20% 的概率保持阴天
  - 如果今天是雨天，明天有 50% 的概率继续下雨，30% 概率变为阴天，20% 概率放晴。
- 我们可以根据今天的状态，预测明天的状态，而不需要知道昨天或者上周的天气。

马尔可夫链 (Markov Chain)

例子：一个简单的天气模型

- 如果今天是晴天，明天有 70% 的概率还是晴天，30% 的概率会阴天。
- 如果今天是阴天，明天有 40% 的概率会放晴，40% 的概率继续下雨，20% 的概率保持阴天
- 如果今天是雨天，明天有 50% 的概率继续下雨，30% 概率变为阴天，20% 概率放晴。

马尔可夫链的局限性

当我们无法直接观测状态时...

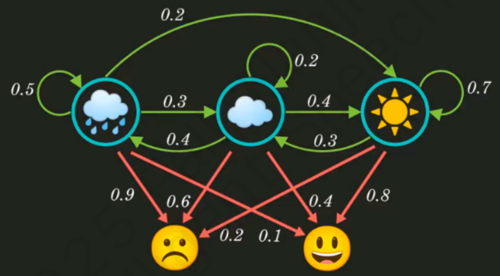
一个猜谜实验

- 想象一下，你在外地，无法直接观测，则原来的天气状态是“隐藏”的。
- 但你有当地的一个朋友，他每天都会出门，并打电话告诉你他今天的心情。
- 他可能的心情有：{开心, 难过}。
- 朋友的心情表现，很大概率是和天气相关的：
  - 晴天时，他更可能 开心。
  - 雨天或者阴天时，他更可能 难过。

任务：根据朋友的心情序列，推断出天气序列

马尔可夫链的局限性

当我们无法直接观测状态时...



$$\begin{bmatrix} 0.5 & 0.3 & 0.2 \\ 0.4 & 0.2 & 0.4 \\ 0.0 & 0.3 & 0.7 \end{bmatrix} \quad \begin{bmatrix} 0.9 & 0.1 \\ 0.6 & 0.4 \\ 0.2 & 0.8 \end{bmatrix}$$

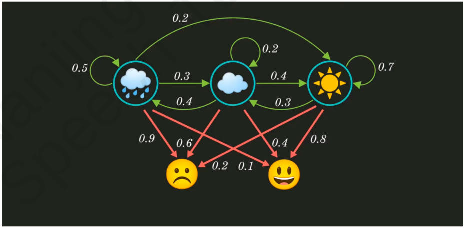
任务：根据朋友的心情序列，推断出天气序列

这个问题引出了一个更强大的模型：隐马尔可夫模型。

- 它包含一条隐藏的、我们无法观测的马尔可夫状态链（例如：天气序列）。
- 它还包含一个可观测的符号序列（例如：朋友的心情序列）。
- 每一个隐藏状态都会以一定的概率“发射”出某个可观测的符号。

HMM 的两大核心要素：

- 状态转移概率：
  隐藏状态之间如何切换？
  (e.g., 从晴天到雨天的概率)
- 发射概率 **(**观测概率**)**:
  在某个隐藏状态下，观测到某个现象的概率是多少？
  (e.g., 在晴天时，心情难过的概率)



HMM 如何应用于语音识别？

建立“天气”和“语音”之间的联系

外地猜天气

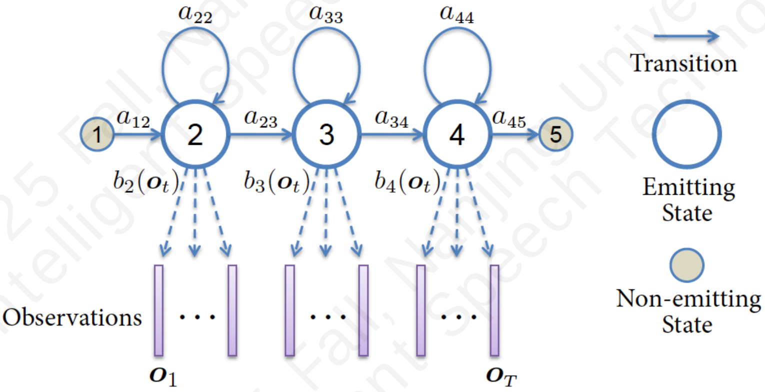
- 隐藏状态：
  天气的真实序列
  (晴 → 晴 → 雨)
- 观测序列：
  朋友的心情序列
  (开心, 开心, 难过)
- 我们的目标：
  根据心情序列，推断出最可能的天气序列。

语音识别系统

- 隐藏状态：
  构成单词的 音素序列
  (/h/ → /e/ → /l/ → /o/)
- 观测序列：
  从语音信号中提取的 声学特征 (MFCC) 序列 $(\vec{o}_1, \vec{o}_2, \vec{o}_3, \ldots, \vec{o}_T)$
- 我们的目标：
  根据声学特征序列，解码出最可能的音素序列，进而得到词序列。

HMM 成为了连接 不可见的语言单元 (音素) 和 可测量的声学信号之间的完美桥梁。

- 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



一个 HMM 模型 λ 由五个要素定义:

- 状态集合 $S = \{s_1, \ldots, s_N\}$: 模型的内部状态，如音素的不同阶段。
- 观测序列 $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$: 每一时刻的声学特征向量[1]。
- 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 $\pi_i = P(q_1 = s_i)$。

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

[1]注意此处的 $\mathbf{X}$ 可以认为是之前提到的语音观测序列 $\mathbf{O}$, 为了跟更广泛的 HMM 的定义保持符号一致，后续将采用 $\mathbf{X}$

1. 评估 (Evaluation): 给定模型 λ，计算观测序列 $\mathbf{X}$ 出现的概率 $p(\mathbf{X}|\lambda)$。
   - 算法: 前向算法 (Forward Algorithm)。

1. 评估 (Evaluation): 给定模型 λ，计算观测序列 $\mathbf{X}$ 出现的概率 $p(\mathbf{X}|\lambda)$。
   - 算法: 前向算法 (Forward Algorithm)。
2. 解码 (Decoding): 给定模型 λ 和观测序列 $\mathbf{X}$，找到最可能的状态序列 Q。
   - 算法: 维特比算法 (Viterbi Algorithm)。

1. 评估 (Evaluation): 给定模型 λ，计算观测序列 $\mathbf{X}$ 出现的概率 $p(\mathbf{X}|\lambda)$。
   - 算法: 前向算法 (Forward Algorithm)。
2. 解码 (Decoding): 给定模型 λ 和观测序列 $\mathbf{X}$，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 $\mathbf{X}$，调整模型参数 λ 使得 $p(\mathbf{X}|\lambda)$ 最大。
   - 算法: Baum-Welch 算法 (即 EM 算法在 HMM 中的特例)。

问题一：评估 (Evaluation)

如何计算 $p(\mathbf{X}|\lambda)$？

目标

给定一个观测序列 $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ 和一个 HMM 模型 $\lambda = (A, B, \pi)$，计算从该模型中观测到序列 $\mathbf{X}$ 的概率 $p(\mathbf{X}|\lambda)$。

- 应用场景:
  - 比如我们训练了两个 HMM，一个代表单词"hello"，一个代表"world"。
  - 给定一段新的语音特征 $\mathbf{X}$，我们可以计算 $p(\mathbf{X}|\lambda_{\text{hello}})$ 和 $p(\mathbf{X}|\lambda_{\text{world}})$。
  - 哪个概率更高，我们就认为这段语音更可能是哪个单词。这是最简单的识别思路。

核心挑战

我们不知道隐藏的状态序列是什么！

- 我们可以穷举所有可能的、长度为 T 的隐藏状态序列 $Q = (q_1, q_2, \dots, q_T)$。
- 对于其中任意一条状态路径 Q，计算它和观测序列 $\mathbf{X}$ 同时发生的联合概率 $p(\mathbf{X}, Q|\lambda)$:

$$p(\mathbf{X}, Q|\lambda) = \pi_{q_1} b_{q_1}(\mathbf{x}_1) \cdot a_{q_1 q_2} b_{q_2}(\mathbf{x}_2) \cdots a_{q_{T-1} q_T} b_{q_T}(\mathbf{x}_T)$$

- 然后，把所有可能路径的概率全部加起来，根据全概率公式得到最终结果:

$$p(\mathbf{X}|\lambda) = \sum_{\text{所有可能的}Q} p(\mathbf{X}, Q|\lambda)$$

为什么不可行？

- 假设有 N 个隐藏状态，序列长度为 T。
- 那么总共有 N^T 条不同的状态路径!
- 这是一个指数级的计算量。对于一段几秒钟的语音（T 可能上千），这是天文数字，计算上完全无法实现。

前向算法：动态规划思想

定义前向变量 $\alpha_t(i)$

定义 (前向变量)

定义前向变量 $\alpha_t(i)$ 为：在给定模型 λ 的情况下，到时刻 t 为止，观测到的部分序列是 $\mathbf{x}_1, \dots, \mathbf{x}_t$，并且在时刻 t 恰好处于状态 s_i 的联合概率。

$$\alpha_t(i) = p(\mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i | \lambda)$$

- 这个变量是算法的基石，它存储了到达某个中间点的“累积概率”。
- 我们的目标是利用 $\alpha_t(\cdot)$ 来递推计算 $\alpha_{t+1}(\cdot)$。

前向算法：详细推导

1. 初始化 (t=1)

我们来计算第一个前向变量 $\alpha_1(i)$。

- 根据定义，$\alpha_1(i) = p(\mathbf{x}_1, q_1 = s_i|\lambda)$。
- 这件事要发生，需要两步：
  1. 首先，初始状态必须是 s_i。这个概率是 π_i。
  2. 接着，在状态 s_i 下，必须发射出观测值 $\mathbf{x}_1$。这个概率是 $b_i(\mathbf{x}_1)$。
- 我们将它们的概率相乘：

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

这为我们的递推提供了起点。我们为所有 N 个状态都计算出这个初始值。

假设我们已经计算出了时刻 t 的所有前向变量 $\alpha_t(1), \alpha_t(2), \ldots, \alpha_t(N)$。如何计算 $\alpha_{t+1}(j)$？

- 要在 $t+1$ 时刻到达状态 s_j，我们必须在 t 时刻处于某个状态 s_i (可以是任意一个)。
- 从 t 时刻的状态 s_i 转移到 $t+1$ 时刻的状态 s_j 的概率是 a_{ij}。
- 到达 t 时刻状态 s_i 的总概率（我们已经算过了！）是 $\alpha_t(i)$。
- 所以，从 t 时刻的某个状态 s_i 出发，在 $t+1$ 时刻到达 s_j 的概率是 $\alpha_t(i)a_{ij}$。
- 把所有可能的上一个状态 s_i $(i = 1, \ldots, N)$ 的情况全部加起来，就得到了到达 $t+1$ 时刻状态 s_j 的总路径概率：$\sum_{i=1}^{N} \alpha_t(i)a_{ij}$。
- 最后，状态 s_j 还要发射出观测值 $\mathbf{x}_{t+1}$，其概率为 $b_j(\mathbf{x}_{t+1})$。

$$\alpha_{t+1}(j) = p(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_{t+1} = s_j)$$

$$\begin{aligned}\alpha_{t+1}(j) &= p(\mathbf{x}_1, \dots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_{t+1} = s_j) \\ &= \sum_{i=1}^{N} p(\mathbf{x}_1, \dots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_t = s_i, q_{t+1} = s_j)\end{aligned}$$

前向算法：详细推导

2. 递推 (数学表达)

$$
\begin{aligned}
\alpha_{t+1}(j) &= p(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_{t+1} \mid \mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j)\, p(\mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j)
\end{aligned}
$$

$$
\begin{aligned}
\alpha_{t+1}(j) &= p(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_{t+1} \mid \mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j)\, p(\mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_{t+1} \mid q_{t+1} = s_j)\, p(\mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j)
\end{aligned}
$$

前向算法：详细推导

2. 递推 (数学表达)

$$
\begin{aligned}
\alpha_{t+1}(j) &= p(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_{t+1} \mid \mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j)\, p(\mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_{t+1} \mid q_{t+1} = s_j)\, p(\mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} b_j(\mathbf{x}_{t+1})\, P(q_{t+1} = s_j \mid \mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = s_i)\, p(\mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = s_i)
\end{aligned}
$$

前向算法：详细推导

2. 递推 (数学表达)

$$
\begin{aligned}
\alpha_{t+1}(j) &= p(\mathbf{x}_1, \dots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_1, \dots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_{t+1} \mid \mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j)\, p(\mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_{t+1} \mid q_{t+1} = s_j)\, p(\mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} b_j(\mathbf{x}_{t+1})\, P(q_{t+1} = s_j \mid \mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i)\, p(\mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i) \\
&= \sum_{i=1}^{N} b_j(\mathbf{x}_{t+1})\, P(q_{t+1} = s_j \mid q_t = s_i)\, \alpha_t(i)
\end{aligned}
$$

$$
\begin{aligned}
\alpha_{t+1}(j) &= p(\mathbf{x}_1, \dots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_1, \dots, \mathbf{x}_t, \mathbf{x}_{t+1}, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_{t+1} \mid \mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j)\, p(\mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} p(\mathbf{x}_{t+1} \mid q_{t+1} = s_j)\, p(\mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i, q_{t+1} = s_j) \\
&= \sum_{i=1}^{N} b_j(\mathbf{x}_{t+1})\, P(q_{t+1} = s_j \mid \mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i)\, p(\mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i) \\
&= \sum_{i=1}^{N} b_j(\mathbf{x}_{t+1})\, P(q_{t+1} = s_j \mid q_t = s_i)\, \alpha_t(i) \\
&= \sum_{i=1}^{N} \alpha_t(i)\, a_{ij}\, b_j(\mathbf{x}_{t+1}) = \left[\sum_{i=1}^{N} \alpha_t(i)\, a_{ij} \right] b_j(\mathbf{x}_{t+1})
\end{aligned}
$$

南京大學 NANJING UNIVERSITY 智能科学与技术学院 School of Intelligence Science and Technology

- 当我们递推到最后一个时刻 T 时，我们得到了一组前向变量 $\alpha_T(1), \alpha_T(2), \ldots, \alpha_T(N)$。
- 回忆 $\alpha_T(i)$ 的定义：$p(\mathbf{x}_1, \ldots, \mathbf{x}_T, q_T = s_i|\lambda)$。
- 这是观测到完整序列 $\mathbf{X}$，并且在 T 时刻以状态 s_i 结尾的概率。
- 我们最终想求的 $p(\mathbf{X}|\lambda)$ 并不关心结尾是哪个状态。所以，我们只需要把所有可能的结尾状态的概率全部加起来即可。

$$\boxed{p(\mathbf{X}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)}$$

效率

每一步递推的计算量大约是 $O(N^2)$。总共需要递推 $T-1$ 次。所以总的计算复杂度是 $O(N^2T)$，这是一个线性时间复杂度，远优于 $O(N^T)$！

目标：计算 $p(\mathbf{X}|\lambda)$

定义前向变量 $\alpha_t(i) = p(\mathbf{x}_1, \dots, \mathbf{x}_t, q_t = s_i|\lambda)$。

1. 初始化:

$$\alpha_1(i) = \pi_i b_i(\mathbf{x}_1), \quad i = 1, \dots, N$$

2. 递推: For $t = 1, \dots, T-1$:

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

3. 终止:

$$p(\mathbf{X}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

问题二：解码 (Decoding)

找到最优的隐藏状态序列

目标

给定观测序列 $\mathbf{X}$ 和模型 λ，找到一条最有可能的隐藏状态序列 $Q^* = (q_1^*, q_2^*, \dots, q_T^*)$。

$$Q^* = \arg\max_{Q} P(Q|\mathbf{X}, \lambda)$$

- 这正是语音识别的核心任务：我们看到了声学特征序列（观测），想知道它背后最有可能的音素序列（隐藏状态）是什么。
- 这个问题和“评估”问题非常相似，但有一个关键区别：
  - 前向算法是把所有路径的概率 加起来 ($\sum$)。
  - 维特比算法是要找到那条概率 最大的路径 (max)。
- 幸运的是，我们同样可以使用动态规划来解决。

南京大学 NANJING UNIVERSITY 智能科学与技术学院 School of Intelligence Science and Technology

定义 (维特比变量 $\delta_t(i)$)

定义 $\delta_t(i)$ 为：到时刻 t 为止，观测到部分序列 $\mathbf{x}_1, \dots, \mathbf{x}_t$，并且在时刻 t 到达状态 s_i 的所有路径中，概率最大的那一条路径的概率。

$$\delta_t(i) = \max_{q_1, \dots, q_{t-1}} p(q_1, \dots, q_{t-1}, q_t = s_i, \mathbf{x}_1, \dots, \mathbf{x}_t | \lambda)$$

定义 (路径记录变量 $\psi_t(i)$)

为了能够最终找出这条路径，我们还需要一个“记事本” $\psi_t(i)$，它记录了在时刻 t 到达状态 s_i 的最优路径，其上一步 **(在 t-1 时刻)** 是哪个状态。

$$\psi_t(i) = \arg\max_{1 \le j \le N} \dots$$

维特比算法：动态规划思想

定义新变量 $\delta_t(i)$ 和 $\psi_t(i)$

$$\alpha_t(i) = p(\mathbf{x}_1, \ldots, \mathbf{x}_t, q_t = s_i)$$

$$\delta_t(i) = \max_{q_1, \ldots, q_{t-1}} p(q_1, \ldots, q_{t-1}, q_t = s_i, \mathbf{x}_1, \ldots, \mathbf{x}_t | \lambda)$$

对比 $\alpha_t(i)$ 和 $\delta_t(i)$

- $\alpha_t(i)$ 是所有到达 (t, s_i) 的路径概率的和。
- $\delta_t(i)$ 是所有到达 (t, s_i) 的路径概率的最大值。

- 对于 $\delta_1(i)$，在 $t=1$ 时刻，路径只有一步。所以不存在“多条路径取最大”的问题。
- 它的计算方式和 $\alpha_1(i)$ 完全一样：

$$\delta_1(i) = \pi_i b_i(\mathbf{x}_1)$$

- 对于路径记录 $\psi_1(i)$，因为 $t=1$ 是起点，没有“上一步”，所以我们将其记为 0。

$$\psi_1(i) = 0$$

假设我们已经计算出了 $t-1$ 时刻的所有 $\delta_{t-1}(i)$。如何计算 $\delta_t(j)$？

- 要找到在 t 时刻到达 s_j 的最优路径，这条路径必然经过 $t-1$ 时刻的某个状态 s_i。
- 对于每一个可能的前驱状态 s_i，从它延伸到 s_j 的路径概率是：

$$\text{路径概率} = \underbrace{\delta_{t-1}(i)}_{\text{到 } s_i \text{ 的最优路径概率}} \times \underbrace{a_{ij}}_{\text{转移概率}}$$

- 我们要从这 N 个可能的前驱状态中，选出使得这个路径概率最大的那一个。

$$\max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij} \right]$$

- 最后，状态 s_j 还要发射出观测值 $\mathbf{x}_t$，所以再乘上 $b_j(\mathbf{x}_t)$。

- 计算最大概率 $\delta_t(j)$:

$$\boxed{\delta_t(j) = \max_{1\leq i\leq N} \left[\delta_{t-1}(i)a_{ij}\right] b_j(\mathbf{x}_t)}$$

- 记录最优路径的前驱状态 $\psi_t(j)$: 我们不仅要知道最大概率是多少，还要记下是哪个 i 让我们得到了这个最大值。

$$\boxed{\psi_t(j) = \arg\max_{1\leq i\leq N} \left[\delta_{t-1}(i)a_{ij}\right]}$$

核心操作

前向算法的核心是 $\sum$，而维特比算法的核心是 max 和 arg max。

- 当递推到最后时刻 T 时，我们计算出了 $\delta_T(1), \dots, \delta_T(N)$。
- 整个序列的最优路径的概率，就是这些值中的最大值。

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

- 最优路径的最后一个状态，就是使 $\delta_T(i)$ 取得最大值的那个状态 i。

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

- 我们已经知道了最优路径的终点 q_T^*。
- 那么路径的上一个点 q_{T-1}^* 是什么呢？答案就记录在我们的“记事本” ψ 中！

$$q_{T-1}^* = \psi_T(q_T^*)$$

- 再往前一步：$q_{T-2}^* = \psi_{T-1}(q_{T-1}^*)$
- ... 以此类推，直到我们回溯到 q_1^*。

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad \text{for } t = T-1, T-2, \dots, 1$$

最终，我们就得到了完整的最优状态序列 $Q^* = (q_1^*, \dots, q_T^*)$。

维特比算法：总结

目标：找到最优状态序列 $Q^* = \arg\max_Q p(Q, \mathbf{X}|\lambda)$

定义 $\delta_t(i)$ 为到时间 t 且处于状态 s_i 的所有路径中的最大概率，$\psi_t(i)$ 为其前驱状态索引。

1. 初始化：

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

2. 递推（$t = 2, \ldots, T$）：

$$\delta_t(j) = \left(\max_{1 \le i \le N} \delta_{t-1}(i) a_{ij} \right) b_j(\mathbf{x}_t), \quad \psi_t(j) = \arg\max_{1 \le i \le N} \delta_{t-1}(i) a_{ij}$$

3. 终止与回溯：

$$P^* = \max_{1 \le i \le N} \delta_T(i), \quad q_T^* = \arg\max_{1 \le i \le N} \delta_T(i), \quad q_t^* = \psi_{t+1}(q_{t+1}^*) \ (t = T-1, \ldots, 1)$$

最终得到最优状态序列 $Q^* = (q_1^*, q_2^*, \ldots, q_T^*)$。

问题三：学习 (Learning)

如何找到最优的模型参数 λ？

给定一个（或多个）观测序列 $\mathbf{X}$，找到一组模型参数 $\lambda = (A, B, \pi)$，使得从该模型生成观测序列 $\mathbf{X}$ 的概率 $p(\mathbf{X}|\lambda)$ 最大。

$$\lambda^* = \arg\max_{\lambda} p(\mathbf{X}|\lambda)$$

- 这是三个问题中最重要的，也是最难的。因为它回答了“如何从数据中训练出一个 HMM 模型”。

问题三：学习 (Learning)

如何找到最优的模型参数 λ？

给定一个（或多个）观测序列 $\mathbf{X}$，找到一组模型参数 $\lambda = (A, B, \pi)$，使得从该模型生成观测序列 $\mathbf{X}$ 的概率 $p(\mathbf{X}|\lambda)$ 最大。

$$\lambda^* = \arg\max_{\lambda} p(\mathbf{X}|\lambda)$$

- 这是三个问题中最重要的，也是最难的。因为它回答了"如何从数据中训练出一个 HMM 模型"。
- 核心困难：这是一个"鸡生蛋，蛋生鸡"的问题。
  - 如果我们知道隐藏状态序列 Q，那么估计参数 λ 将会非常简单（只需要统计频率即可）。
  - 如果我们知道模型参数 λ，我们就可以推断隐藏状态序列 Q（例如用维特比算法）。

给定一个（或多个）观测序列 $\mathbf{X}$，找到一组模型参数 $\lambda = (A, B, \pi)$，使得从该模型生成观测序列 $\mathbf{X}$ 的概率 $p(\mathbf{X}|\lambda)$ 最大。

$$\lambda^* = \arg\max_{\lambda} p(\mathbf{X}|\lambda)$$

- 这是三个问题中最重要的，也是最难的。因为它回答了“如何从数据中训练出一个 HMM 模型”。
- 核心困难：这是一个“鸡生蛋，蛋生鸡”的问题。
  - 如果我们知道隐藏状态序列 Q，那么估计参数 λ 将会非常简单（只需要统计频率即可）。
  - 如果我们知道模型参数 λ，我们就可以推断隐藏状态序列 Q（例如用维特比算法）。
- 但现实是，我们两者都不知道！我们只有观测序列 $\mathbf{X}$。

解决方案

使用一种迭代算法——Baum-Welch 算法，它是著名的期望最大化 (Expectation-Maximization, EM) 算法在 HMM 上的一个特例。

学习问题：一个理想化的场景

如果我们有“上帝视角”（知道隐藏状态）

设想

假设除了观测序列 $\mathbf{X}$ 外，我们还拥有了其对应的、真实的隐藏状态序列 Q。这种包含隐藏信息的数据被称为"完整数据"。参数估计就退化成了简单的频率统计

- 初始概率 π_i:

$$\hat{\pi}_i = \frac{\text{序列以状态}s_i\text{开头的次数}}{\text{总序列数}}$$

学习问题：一个理想化的场景

如果我们有“上帝视角”（知道隐藏状态）

设想

假设除了观测序列 $\mathbf{X}$ 外，我们还拥有了其对应的、真实的隐藏状态序列 Q。这种包含隐藏信息的数据被称为"完整数据"。参数估计就退化成了简单的频率统计

- 初始概率 π_i：

$$\hat{\pi}_i = \frac{\text{序列以状态}s_i\text{开头的次数}}{\text{总序列数}}$$

- 转移概率 a_{ij}：

$$\hat{a}_{ij} = \frac{\text{从状态}s_i\text{转移到}s_j\text{的次数}}{\text{从状态}s_i\text{出发的所有转移的总次数}}$$

学习问题：一个理想化的场景

如果我们有“上帝视角”（知道隐藏状态）

设想

假设除了观测序列 $\mathbf{X}$ 外，我们还拥有了其对应的、真实的隐藏状态序列 Q。这种包含隐藏信息的数据被称为"完整数据"。参数估计就退化成了简单的频率统计

- 初始概率 π_i:

$$\hat{\pi}_i = \frac{\text{序列以状态}s_i\text{开头的次数}}{\text{总序列数}}$$

- 转移概率 a_{ij}:

$$\hat{a}_{ij} = \frac{\text{从状态}s_i\text{转移到}s_j\text{的次数}}{\text{从状态}s_i\text{出发的所有转移的总次数}}$$

- 发射概率 $b_j(k)$ (以离散观测为例):

$$\hat{b}_j(k) = \frac{\text{在状态}s_j\text{时观测到符号}v_k\text{的次数}}{\text{处于状态}s_j\text{的总次数}}$$

学习问题：一个理想化的场景

如果我们有“上帝视角”（知道隐藏状态）

设想

假设除了观测序列 $\mathbf{X}$ 外，我们还拥有了其对应的、真实的隐藏状态序列 Q。这种包含隐藏信息的数据被称为"完整数据"。参数估计就退化成了简单的频率统计

- 初始概率 π_i:

$$\hat{\pi}_i = \frac{\text{序列以状态}s_i\text{ 开头的次数}}{\text{总序列数}}$$

- 转移概率 a_{ij}:

$$\hat{a}_{ij} = \frac{\text{从状态}s_i\text{ 转移到}s_j\text{ 的次数}}{\text{从状态}s_i\text{ 出发的所有转移的总次数}}$$

- 发射概率 $b_j(k)$ (以离散观测为例):

$$\hat{b}_j(k) = \frac{\text{在状态}s_j\text{ 时观测到符号}v_k\text{ 的次数}}{\text{处于状态}s_j\text{ 的总次数}}$$

EM 算法的核心直觉：猜想与优化

既然不能直接“计数”，那我们可以“软计数”——即计算期望次数。

E-Step (期望)

- 固定当前的模型参数 λ。
- 基于这个模型和观测 X，我们不再猜测唯一的隐藏状态序列，而是计算出所有我们关心的事件的“期望”：
  - 在时刻 t 处于状态 s_i 的概率。
  - 在时刻 t 从 s_i 转移到 s_j 的概率。
- 这些概率可以看作是“软计数”或“期望计数”。

M-Step (最大化)

- 假设 E-Step 计算出的“期望计数”就是真实的、准确的计数。
- 使用上一页的频率统计公式，只不过把“硬计数”替换为 E-Step 得到的“期望计数”，来重新估计一组新的、更好的参数 λ^{new}。

$$\text{初始化} \xrightarrow{\text{E-Step}} \text{期望} \xrightarrow{\text{M-Step}} \text{新参数} \xrightarrow{\text{E-Step}} \text{新期望} \xrightarrow{\text{M-Step}} \dots$$

不断迭代，直到模型参数收敛。

E-Step 的工具：后向变量 $\beta_t(i)$

为了计算期望，除了前向变量 $\alpha_t(i)$，我们还需要一个它的“镜像”——后向变量 $\beta_t(i)$。

定义 (后向变量)

定义后向变量 $\beta_t(i)$ 为：在给定模型 λ 并且在时刻 t 处于状态 s_i 的条件下，观测到从 $t+1$ 到结尾的部分序列 $\mathbf{x}_{t+1}, \dots, \mathbf{x}_T$ 的概率。

$$\beta_t(i) = p(\mathbf{x}_{t+1}, \dots, \mathbf{x}_T | q_t = s_i, \lambda)$$

- **初始化 ($t = T$)**: 在结尾时刻 T，后面没有序列了，所以我们定义 $\beta_T(i) = 1$。

E-Step 的工具：后向变量 $\beta_t(i)$

为了计算期望，除了前向变量 $\alpha_t(i)$，我们还需要一个它的“镜像”——后向变量 $\beta_t(i)$。

定义 (后向变量)

定义后向变量 $\beta_t(i)$ 为：在给定模型 λ 并且在时刻 t 处于状态 s_i 的条件下，观测到从 $t+1$ 到结尾的部分序列 $\mathbf{x}_{t+1}, \ldots, \mathbf{x}_T$ 的概率。

$$\beta_t(i) = p(\mathbf{x}_{t+1}, \ldots, \mathbf{x}_T | q_t = s_i, \lambda)$$

- 初始化 ($t = T$): 在结尾时刻 T，后面没有序列了，所以我们定义 $\beta_T(i) = 1$。
- 递推 (从 $t+1$ 到 t):
  - 要计算 $\beta_t(i)$，需要考虑从状态 s_i 出发，在 $t+1$ 时刻可能到达的所有状态 s_j。
  - 对于每一个 s_j，路径的概率是：a_{ij} (转移) $\times$ $b_j(\mathbf{x}_{t+1})$ (发射) $\times$ $\beta_{t+1}(j)$ (后续路径)。
  - 把所有可能的 s_j 的情况加起来。

$$\boxed{\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j)}$$

有了 $\alpha_t(i)$ 和 $\beta_t(i)$，我们就可以计算第一个关键期望值 $\gamma_t(i)$。

定义 (状态占用概率 $\gamma_t(i)$)

给定模型 λ 和整个观测序列 $\mathbf{X}$，在时刻 t 处于状态 s_i 的概率。

$$\gamma_t(i) = P(q_t = s_i|\mathbf{X}, \lambda)$$

- 直觉：序列在 t 时刻经过 s_i 这件事，可以被 $\alpha_t(i)$ 和 $\beta_t(i)$ “夹在中间”。
  - $\alpha_t(i)$ 是从头到 (t, s_i) 的概率。
  - $\beta_t(i)$ 是从 (t, s_i) 到结尾的概率。
  - 两者相乘 $\alpha_t(i)\beta_t(i)$ 就正比于经过 (t, s_i) 的所有路径的概率总和。

E-Step：计算期望 (1/2)

在时刻 t 处于状态 s_i 的概率

有了 $\alpha_t(i)$ 和 $\beta_t(i)$，我们就可以计算第一个关键期望值 $\gamma_t(i)$。

定义 (状态占用概率 $\gamma_t(i)$)

给定模型 λ 和整个观测序列 $\mathbf{X}$，在时刻 t 处于状态 s_i 的概率。

$$\gamma_t(i) = P(q_t = s_i|\mathbf{X}, \lambda)$$

- 直觉：序列在 t 时刻经过 s_i 这件事，可以被 $\alpha_t(i)$ 和 $\beta_t(i)$ “夹在中间”。
  - $\alpha_t(i)$ 是从头到 (t, s_i) 的概率。
  - $\beta_t(i)$ 是从 (t, s_i) 到结尾的概率。
  - 两者相乘 $\alpha_t(i)\beta_t(i)$ 就正比于经过 (t, s_i) 的所有路径的概率总和。
- 公式：我们用总概率 $p(\mathbf{X}|\lambda)$ 进行归一化，就得到条件概率。

$$\gamma_t(i) = \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_T, q_t = s_i|\lambda)}{p(\mathbf{x}_1, \dots, \mathbf{x}_T|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

我们还需要计算第二个关键期望值 $\xi_t(i,j)$。

定义 (转移概率 $\xi_t(i,j)$)

给定模型 λ 和整个观测序列 $\mathbf{X}$，在时刻 t 处于状态 s_i 并且在时刻 $t+1$ 处于状态 s_j 的联合概率。

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | \mathbf{X}, \lambda)$$

- 直觉: 这次我们需要把 $t \to t+1$ 的一步转移 “夹在中间”。
  - 从头到 (t, s_i): $\alpha_t(i)$
  - 从 (t, s_i) 转移到 $(t+1, s_j)$ 并发射 $\mathbf{x}_{t+1}$: $a_{ij} b_j(\mathbf{x}_{t+1})$
  - 从 $(t+1, s_j)$ 到结尾: $\beta_{t+1}(j)$

我们还需要计算第二个关键期望值 $\xi_t(i,j)$。

定义 (转移概率 $\xi_t(i,j)$)

给定模型 λ 和整个观测序列 $\mathbf{X}$，在时刻 t 处于状态 s_i 并且在时刻 $t+1$ 处于状态 s_j 的联合概率。

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | \mathbf{X}, \lambda)$$

- 直觉：这次我们需要把 $t \to t+1$ 的一步转移“夹在中间”。
  - 从头到 (t, s_i): $\alpha_t(i)$
  - 从 (t, s_i) 转移到 $(t+1, s_j)$ 并发射 $\mathbf{x}_{t+1}$: $a_{ij} b_j(\mathbf{x}_{t+1})$
  - 从 $(t+1, s_j)$ 到结尾: $\beta_{t+1}(j)$
- 公式：把它们全部乘起来，再用总概率 $p(\mathbf{X}|\lambda)$ 进行归一化。

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j)}{p(\mathbf{X}|\lambda)}$$

M-Step：最大化期望，重估参数

用“期望计数”代替“真实计数”

在 E-Step 得到 $\gamma_t(i)$ 和 $\xi_t(i,j)$ 后，我们就可以进行 M-Step 来更新参数了。

- 期望初始状态为 s_i 的次数 $\rightarrow \gamma_1(i)$
- 期望从 s_i 出发的总次数 $\rightarrow \sum_{t=1}^{T-1} \gamma_t(i)$
- 期望从 s_i 转移到 s_j 的总次数 $\rightarrow \sum_{t=1}^{T-1} \xi_t(i,j)$
- 期望在状态 s_j 的总次数 $\rightarrow \sum_{t=1}^{T} \gamma_t(j)$

M-Step：最大化期望，重估参数

用“期望计数”代替“真实计数”

在 E-Step 得到 $\gamma_t(i)$ 和 $\xi_t(i,j)$ 后，我们就可以进行 M-Step 来更新参数了。

- 期望初始状态为 s_i 的次数 $\rightarrow \gamma_1(i)$
- 期望从 s_i 出发的总次数 $\rightarrow \sum_{t=1}^{T-1} \gamma_t(i)$
- 期望从 s_i 转移到 s_j 的总次数 $\rightarrow \sum_{t=1}^{T-1} \xi_t(i,j)$
- 期望在状态 s_j 的总次数 $\rightarrow \sum_{t=1}^{T} \gamma_t(j)$

参数重估公式

$$\hat{\pi}_i = \text{期望初始在} s_i \text{ 的概率} = \gamma_1(i)$$

$$\hat{a}_{ij} = \frac{\text{期望从} s_i \rightarrow s_j \text{ 的次数}}{\text{期望从} s_i \text{ 出发的次数}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\hat{b}_j(k) = \frac{\text{期望在} s_j \text{ 且观测为} v_k \text{ 的次数}}{\text{期望在} s_j \text{ 的次数}} = \frac{\sum_{t=1, \mathbf{x}_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

目标：找到 $\lambda = \arg\max p(\mathbf{X}|\lambda)$

通过 EM 迭代算法实现。

1. 初始化:

- 随机（或根据先验知识）初始化模型参数 $\lambda = (A, B, \pi)$。

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

目标：找到 $\lambda = \arg\max p(\mathbf{X}|\lambda)$

通过 EM 迭代算法实现。

1. 初始化:
   - 随机（或根据先验知识）初始化模型参数 $\lambda = (A, B, \pi)$。
2. 循环迭代直到收敛:
   - **E-Step**:
     1. 使用当前 λ 和观测 $\mathbf{X}$，运行前向算法计算所有 $\alpha_t(i)$。
     2. 使用当前 λ 和观测 $\mathbf{X}$，运行后向算法计算所有 $\beta_t(i)$。
     3. 计算期望值 $\gamma_t(i)$ 和 $\xi_t(i,j)$。

目标：找到 $\lambda = \arg\max p(\mathbf{X}|\lambda)$

通过 EM 迭代算法实现。

1. 初始化:
   - 随机（或根据先验知识）初始化模型参数 $\lambda = (A, B, \pi)$。
2. 循环迭代直到收敛:
   - **E-Step**:
     1. 使用当前 λ 和观测 $\mathbf{X}$，运行前向算法计算所有 $\alpha_t(i)$。
     2. 使用当前 λ 和观测 $\mathbf{X}$，运行后向算法计算所有 $\beta_t(i)$。
     3. 计算期望值 $\gamma_t(i)$ 和 $\xi_t(i, j)$。
   - **M-Step**:
     1. 使用 $\gamma_t(i)$ 和 $\xi_t(i, j)$，根据重估公式计算新的参数 $\lambda^{\text{new}} = (\hat{A}, \hat{B}, \hat{\pi})$。

目标：找到 $\lambda = \arg\max p(\mathbf{X}|\lambda)$

通过 EM 迭代算法实现。

1. 初始化:
   - 随机（或根据先验知识）初始化模型参数 $\lambda = (A, B, \pi)$。
2. 循环迭代直到收敛:
   - **E-Step**:
     1. 使用当前 λ 和观测 $\mathbf{X}$，运行前向算法计算所有 $\alpha_t(i)$。
     2. 使用当前 λ 和观测 $\mathbf{X}$，运行后向算法计算所有 $\beta_t(i)$。
     3. 计算期望值 $\gamma_t(i)$ 和 $\xi_t(i, j)$。
   - **M-Step**:
     1. 使用 $\gamma_t(i)$ 和 $\xi_t(i, j)$，根据重估公式计算新的参数 $\lambda^{\text{new}} = (\hat{A}, \hat{B}, \hat{\pi})$。
   - 更新:
     - $\lambda \leftarrow \lambda^{\text{new}}$

Baum-Welch 算法：总结

目标：找到 $\lambda = \arg\max p(\mathbf{X}|\lambda)$

通过 EM 迭代算法实现。

1. 初始化:
   - 随机（或根据先验知识）初始化模型参数 $\lambda = (A, B, \pi)$。
2. 循环迭代直到收敛:
   - **E-Step**:
     1. 使用当前 λ 和观测 $\mathbf{X}$，运行前向算法计算所有 $\alpha_t(i)$。
     2. 使用当前 λ 和观测 $\mathbf{X}$，运行后向算法计算所有 $\beta_t(i)$。
     3. 计算期望值 $\gamma_t(i)$ 和 $\xi_t(i,j)$。
   - **M-Step**:
     1. 使用 $\gamma_t(i)$ 和 $\xi_t(i,j)$，根据重估公式计算新的参数 $\lambda^{\text{new}} = (\hat{A}, \hat{B}, \hat{\pi})$。
   - 更新:
     - $\lambda \leftarrow \lambda^{\text{new}}$

每一次迭代都保证 $p(\mathbf{X}|\lambda^{\text{new}}) \geq p(\mathbf{X}|\lambda)$，因此算法最终会收敛到一个局部最优解

- 在前面的例子中，我们有时会假设观测是离散的（如 v_k）。但在语音识别中，声学特征（如 MFCC）是连续的、高维的向量 $\mathbf{x}_t \in \mathbb{R}^D$。

为什么需要 GMM？

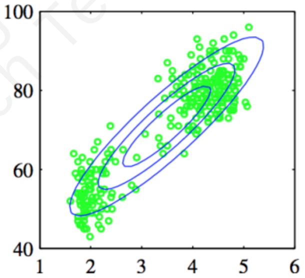
从离散观测到连续观测

- 在前面的例子中，我们有时会假设观测是离散的（如 v_k）。但在语音识别中，声学特征（如 MFCC）是连续的、高维的向量 $\mathbf{x}_t \in \mathbb{R}^D$。
- 因此，发射概率 $b_j(\mathbf{x}_t) = p(\mathbf{x}_t | q_t = s_j)$ 必须是一个能够对连续向量进行建模的概率密度函数 **(PDF)**。

- 在前面的例子中，我们有时会假设观测是离散的（如 v_k）。但在语音识别中，声学特征（如 MFCC）是连续的、高维的向量 $\mathbf{x}_t \in \mathbb{R}^D$。
- 因此，发射概率 $b_j(\mathbf{x}_t) = p(\mathbf{x}_t | q_t = s_j)$ 必须是一个能够对连续向量进行建模的概率密度函数 **(PDF)**。
- 最简单的选择是多维高斯分布 $\mathcal{N}(\mathbf{x}|\mu, \Sigma)$。但它存在一个严重问题：

单高斯分布的局限性

单一高斯分布的建模能力太有限，无法捕捉复杂、非对称、多峰的声学特征分布。



解决方案：高斯混合模型 (GMM)

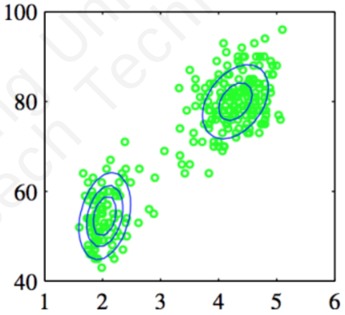
- 核心思想：使用多个高斯分布的加权和来共同拟合一个复杂的分布。
- 万能逼近器：高斯混合模型 (GMM) 是一个“万能逼近器”(Universal Approximator)，理论上只要分量足够多，它可以拟合任意形状的连续概率分布。

GMM-HMM 的发射概率

状态 j 的发射概率 $b_j(\mathbf{x})$ 由 M 个高斯分量混合而成：

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

每个 HMM 状态 s_j 都拥有自己的一套 GMM 参数：$\{w_{jk}, \mu_{jk}, \Sigma_{jk}\}_{k=1}^{M}$。



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的连续概率分布。

GMM 公式

状态 j 的发射概率由 M 个高斯分量混合而成：

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

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

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

如何调整 Baum-Welch 算法？

- 我们需要再次使用 EM 算法来估计 GMM-HMM 的参数。
- 好消息是，对于 HMM 的宏观参数 π 和 A (转移矩阵)，其重估公式保持不变！因为它们只跟状态序列有关，与状态内部如何发射观测无关。

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

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

如何调整 Baum-Welch 算法？

- 我们需要再次使用 EM 算法来估计 GMM-HMM 的参数。
- 好消息是，对于 HMM 的宏观参数 π 和 A (转移矩阵)，其重估公式保持不变！因为它们只跟状态序列有关，与状态内部如何发射观测无关。

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

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

- 真正的挑战在于 **M-Step** 中对发射概率 B 的更新。
  - 以前我们是更新一个简单的概率表 $b_j(k)$。
  - 现在，我们需要更新每个状态 s_j 内部的 GMM 参数：权重 w_{jk}，均值 μ_{jk}，和协方差 Σ_{jk}。

核心问题

为了更新第 j 个状态的第 k 个高斯分量，我们需要知道：在所有被认为是“属于”状态 j 的观测中，哪些观测又应该“属于”这个高斯分量 k？

为了解决上述问题，我们需要计算一个更精细的期望值。

定义 (分量占用概率 $\gamma_t(j,k)$)

给定模型 λ 和观测序列 $\mathbf{X}$，在时刻 t 处于状态 s_j，并且观测 $\mathbf{x}_t$ 是由状态 s_j 的第 k 个高斯分量所生成的联合概率。

$$\gamma_t(j,k) = P(q_t = s_j, k_t = k|\mathbf{X}, \lambda)$$

其中 $k_t = k$ 表示在 t 时刻选中了第 k 个高斯分量。

为了解决上述问题，我们需要计算一个更精细的期望值。

定义 (分量占用概率 $\gamma_t(j,k)$)

给定模型 λ 和观测序列 $\mathbf{X}$，在时刻 t 处于状态 s_j，并且观测 $\mathbf{x}_t$ 是由状态 s_j 的第 k 个高斯分量所生成的联合概率。

$$\gamma_t(j,k) = P(q_t = s_j, k_t = k|\mathbf{X}, \lambda)$$

其中 $k_t = k$ 表示在 t 时刻选中了第 k 个高斯分量。

如何计算？我们可以利用已知的 $\gamma_t(j) = P(q_t = s_j|\mathbf{X}, \lambda)$：

$$\begin{aligned}
\gamma_t(j,k) &= P(k_t = k|q_t = s_j, \mathbf{X}, \lambda) \cdot P(q_t = s_j|\mathbf{X}, \lambda) \\
&= \left(\frac{p(\mathbf{x}_t|q_t = s_j, k_t = k)P(k_t = k|q_t = s_j)}{p(\mathbf{x}_t|q_t = s_j)}\right) \cdot \gamma_t(j) \\
&= \left(\frac{w_{jk}\mathcal{N}(\mathbf{x}_t|\mu_{jk}, \Sigma_{jk})}{\sum_{m=1}^{M} w_{jm}\mathcal{N}(\mathbf{x}_t|\mu_{jm}, \Sigma_{jm})}\right) \cdot \gamma_t(j)
\end{aligned}$$

GMM-HMM 的 M-Step：重估 GMM 参数

用“期望贡献度”进行加权平均

有了 $\gamma_t(j,k)$，我们就可以把它作为权重，来更新 GMM 的参数。这和标准 GMM 的 EM 算法非常相似。

更新混合权重 w_{jk}

新权重是分量 (j,k) 的总期望贡献度，除以状态 j 的总期望贡献度。

$$\hat{w}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{m=1}^{M} \gamma_t(j,m)} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \gamma_t(j)}$$

GMM-HMM 的 M-Step：重估 GMM 参数

用“期望贡献度”进行加权平均

有了 $\gamma_t(j,k)$，我们就可以把它作为权重，来更新 GMM 的参数。这和标准 GMM 的 EM 算法非常相似。

更新均值 μ_{jk}

新均值是所有观测向量 $\mathbf{x}_t$ 的加权平均，权重为 $\gamma_t(j,k)$。

$$\hat{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)\mathbf{x}_t}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

有了 $\gamma_t(j,k)$，我们就可以把它作为权重，来更新 GMM 的参数。这和标准 GMM 的 EM 算法非常相似。

更新协方差 Σ_{jk}

新协方差是每个观测向量与新均值之差的平方的加权平均。

$$\hat{\Sigma}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)(\mathbf{x}_t - \hat{\mu}_{jk})(\mathbf{x}_t - \hat{\mu}_{jk})^\top}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

单音素 (Monophone) HMM 的局限性

- 到目前为止，我们讨论的 HMM 模型（如为每个音素 /ah/, /b/, /p/ ... 建立一个 HMM）是上下文无关 **(Context-Independent)** 的。
- 这意味着，音素 /ah/ 的发音模型，在任何情况下都是完全一样的。
- 现实情况: 发音会受到协同发音 (Coarticulation) 的严重影响。

例如，音素 /ah/ 的发音

- 在单词"pub" (p-**ah**-b) 中的发音
- 在单词"tub" (t-**ah**-b) 中的发音
- 在单词"fuss" (f-**ah**-s) 中的发音

这三个 /ah/ 的声学实现（即语音特征）是截然不同的，因为它们受到了前后音素的影响。

结论

单音素模型无法捕捉这种上下文相关性，导致模型精度不足。

解决方案：三音素 (Triphone) HMM

核心思想

为每一个“上下文相关”的音素建立一个独立的 HMM 模型，最常见的就是三音素。

定义 (三音素)

一个三音素由中心音素及其左、右相邻的音素共同定义。

$$L-C+R$$

其中 L 是左邻音素，C 是中心音素，R 是右邻音素。

例如

单词"pub" (/p/ /ah/ /b/) 中，音素 /ah/ 对应的三音素模型是 p-ah+b。
而单词"fuss" (/f/ /ah/ /s/) 中，音素 /ah/ 对应的三音素模型是 f-ah+s。
这样，*p-ah+b* 和 *f-ah+s* 就可以拥有各自独立的 *HMM* 参数，从而精确地描述它们在不同上下文中的发音。

三音素模型带来的新问题

(1): 参数爆炸

- 假设一个语言有 40 个基本音素。
- 理论上，可能存在的三音素数量为：

$$40(\text{左}) \times 40(\text{中}) \times 40(\text{右}) = 40^3 = 64,000 \text{ 个}$$

- 如果每个三音素 HMM 有 3 个状态，每个状态的 GMM 需要均值、方差等参数...
- 这会导致模型参数的总量急剧膨胀，变得异常庞大！

问题一：参数爆炸 (Parameter Explosion)

模型参数过多，不仅需要巨大的存储空间，也使得训练变得极其困难和耗时。

三音素模型带来的新问题

(2): 数据稀疏

- 在一个非常大的语音训练数据库中，许多理论上可能存在的三音素，实际上可能从未出现过，或者只出现过一两次。
- 例如，像 `z-iy+q` 这样的组合在英语中可能永远不会出现。
- 如果一个三音素在训练数据中没有出现，我们就无法为它训练模型！
- 即使只出现几次，这点数据也完全不足以训练出一个稳定、可靠的 GMM 参数。

问题二：数据稀疏 (Data Sparsity)

训练数据覆盖不全，导致大量三音素模型无法被有效训练。这在测试时会遇到严重的“未登录三音素”问题。

我们需要一个 **trade-off**！

Trade-off：状态绑定/聚类

核心思想

我们既不希望像单音素那样“所有上下文都共享参数”，也不希望像三音素那样“所有上下文都独立参数”。我们希望：声学特性相似的上下文，可以共享参数。

- 直觉: 考虑来自"pub" 的三音素 p-ah+b 和来自"tub" 的 t-ah+b。
  - 它们的中心音素都是 /ah/。
  - 虽然左邻音素不同 (/p/ vs /t/)，但它们都是清塞音 **(voiceless plosive)**，且右邻音素 /b/ 完全相同。
  - 因此，它们对中心音素 /ah/ 的声学影响被认为是非常相似的。
- 解决方案: 我们可以让 p-ah+b 和 t-ah+b 这两个三音素模型的对应状态共享同一套 GMM 参数。
- 这就是状态绑定 **(State Tying)** 或 状态聚类 **(State Clustering)**。

这样做的好处

1. 减少参数量: 多个状态共享一套参数，总参数量大大减少。

我们不能凭感觉去绑定状态。我们需要一个自动的、数据驱动的方法来决定。

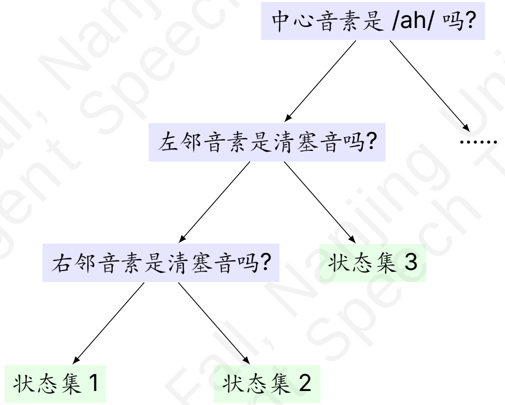
目标

找到一个最优的聚类方案，使得聚类后的模型在训练集上的似然概率 (Likelihood) 最大。

- 方法：对每个单音素（如 /ah/）的每个 HMM 状态，分别进行聚类。
- 例如：我们把所有中心音素是 /ah/ 的三音素（如来自"pub" 的 p-ah+b，来自"tub" 的 t-ah+b，来自"fuss" 的 f-ah+s 等）的第 1 个状态都拿出来，放在一个集合里，然后试图对它们进行聚类。
- 如何聚类？通过提出一系列关于上下文音素的“是/非”问题，来逐步地、层层地划分这些状态。这自然就引出了——决策树 **(Decision Tree)**。

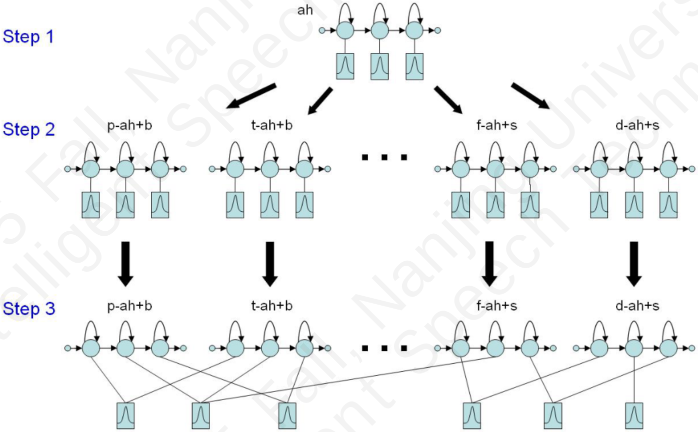
核心思想

将声学特性相似的三音素状态“绑定”在一起，共享 GMM 参数。



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。



GMM-HMM 的成就与局限

在深度学习兴起之前，GMM-HMM 是语音识别领域绝对的霸主，统治了数十年。但它也存在一些难以突破的理论瓶颈。

GMM-HMM 的成就与局限

在深度学习兴起之前，GMM-HMM 是语音识别领域绝对的霸主，统治了数十年。但它也存在一些难以突破的理论瓶颈。

- 建模能力有限：GMM 是一个“浅层”模型，它在拟合声学特征和 HMM 状态之间复杂的、高度非线性的关系时能力不足。

GMM-HMM 的成就与局限

在深度学习兴起之前，GMM-HMM 是语音识别领域绝对的霸主，统治了数十年。但它也存在一些难以突破的理论瓶颈。

- 建模能力有限: GMM 是一个“浅层”模型，它在拟合声学特征和 HMM 状态之间复杂的、高度非线性的关系时能力不足。
- 特征相关性假设: GMM 假设输入特征向量的各个维度是（或近似是）不相关的。这迫使我们必须使用经过复杂处理的特征（如 MFCC），而不是更原始的特征。

GMM-HMM 的成就与局限

在深度学习兴起之前，GMM-HMM 是语音识别领域绝对的霸主，统治了数十年。但它也存在一些难以突破的理论瓶颈。

- 建模能力有限：GMM 是一个“浅层”模型，它在拟合声学特征和 HMM 状态之间复杂的、高度非线性的关系时能力不足。
- 特征相关性假设：GMM 假设输入特征向量的各个维度是（或近似是）不相关的。这迫使我们必须使用经过复杂处理的特征（如 MFCC），而不是更原始的特征。
- 上下文信息缺失：GMM 对每一帧 $\mathbf{x}_t$ 进行独立建模，完全忽略了相邻帧 $\mathbf{x}_{t-1}, \mathbf{x}_{t+1}$ 中包含的大量上下文信息。这违背了语音信号的短时平稳特性。

GMM-HMM 的成就与局限

在深度学习兴起之前，GMM-HMM 是语音识别领域绝对的霸主，统治了数十年。但它也存在一些难以突破的理论瓶颈。

- 建模能力有限：GMM 是一个“浅层”模型，它在拟合声学特征和 HMM 状态之间复杂的、高度非线性的关系时能力不足。
- 特征相关性假设：GMM 假设输入特征向量的各个维度是（或近似是）不相关的。这迫使我们必须使用经过复杂处理的特征（如 MFCC），而不是更原始的特征。
- 上下文信息缺失：GMM 对每一帧 $\mathbf{x}_t$ 进行独立建模，完全忽略了相邻帧 $\mathbf{x}_{t-1}, \mathbf{x}_{t+1}$ 中包含的大量上下文信息。这违背了语音信号的短时平稳特性。

核心问题

GMM 作为发射概率模型，已经成为了整个系统的性能瓶颈。我们能否找到一个更强大的模型来替代它？

混合系统的诞生：DNN-HMM

核心思想

各取所长：保留 HMM 强大的时序建模能力（状态转移），同时用一个强大的深度神经网络 **(DNN)** 来替换掉相对较弱的 GMM，用以计算发射概率。

$$\texttt{GMM-HMM: HMM}(\pi, A) + \texttt{GMM}(B)$$

$$\Downarrow$$

$$\texttt{DNN-HMM: HMM}(\pi, A) + \texttt{DNN}(B)$$

核心思想

各取所长：保留 HMM 强大的时序建模能力（状态转移），同时用一个强大的深度神经网络 **(DNN)** 来替换掉相对较弱的 GMM，用以计算发射概率。

$$\texttt{GMM-HMM:}\ \texttt{HMM}(\pi, A)\ \texttt{+}\ \texttt{GMM}(B)$$
$$\Downarrow$$
$$\texttt{DNN-HMM:}\ \texttt{HMM}(\pi, A)\ \texttt{+}\ \texttt{DNN}(B)$$

- 在这个混合系统中，DNN 的任务是：给定一个声学特征向量 $\mathbf{x}_t$，计算出它属于每一个 HMM 状态 s_j 的概率。
- HMM 的任务保持不变：使用维特比算法，在给定发射概率（由 DNN 提供）和转移概率的情况下，寻找最优的状态序列。

- **HMM 需要什么？**
  HMM 解码器需要的是似然概率 $p(\mathbf{x}_t|s_j)$，即在状态 s_j 下生成观测 $\mathbf{x}_t$ 的概率。

- **HMM** 需要什么？
  HMM 解码器需要的是似然概率 $p(\mathbf{x}_t|s_j)$，即在状态 s_j 下生成观测 $\mathbf{x}_t$ 的概率。
- **DNN** 提供什么？
  DNN 本质上是一个分类器。它的输出（经过 Softmax 层后）是后验概率 $P(s_j|\mathbf{x}_t)$，即给定观测 $\mathbf{x}_t$ 后，当前帧属于状态 s_j 的概率。

如何从后验转换到似然？

我们可以使用贝叶斯定理：

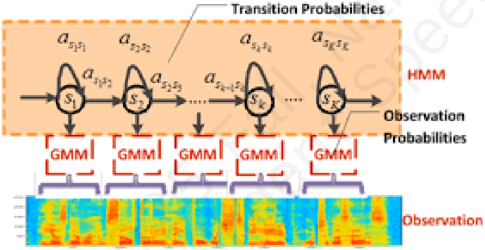
$$p(\mathbf{x}_t|s_j) = \frac{P(s_j|\mathbf{x}_t)p(\mathbf{x}_t)}{P(s_j)}$$

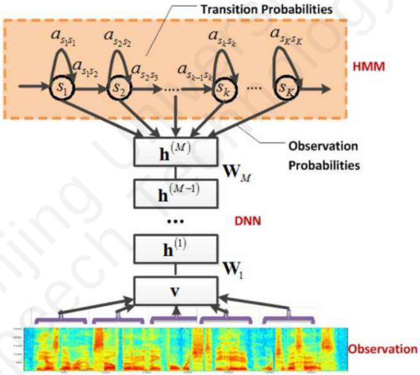
在解码的任意时刻 t，对于所有状态 j 来说，$p(\mathbf{x}_t)$ 都是一个常数，可以在比较中被忽略。因此，我们实际使用的" 伪似然" 是：

$$\text{pseudo-likelihood} \propto \frac{P(s_j|\mathbf{x}_t)}{P(s_j)}$$

- $P(s_j|\mathbf{x}_t)$：由 **DNN** 网络输出。
- $P(s_j)$：状态 s_j 的先验概率，可以通过它在训练数据中出现的频率来估计。

GMM-HMM v.s. DNN-HMM





这是一个多阶段的过程，需要一个已有的 GMM-HMM 系统来“引导”DNN 的训练。

1. 训练一个 **GMM-HMM** 系统：
   首先，使用我们前面介绍的 Baum-Welch (ML) 和判别式训练 (MMI/MPE) 方法，训练一个尽可能好的 GMM-HMM 系统。

这是一个多阶段的过程，需要一个已有的 GMM-HMM 系统来“引导”DNN 的训练。

1. 训练一个 **GMM-HMM** 系统:
   首先，使用我们前面介绍的 Baum-Welch (ML) 和判别式训练 (MMI/MPE) 方法，训练一个尽可能好的 GMM-HMM 系统。
2. 生成“强制对齐”**(Forced Alignment)**:
   使用上一步训练好的 GMM-HMM，对所有的训练语音和其对应的正确文本进行维特比解码。这会为训练数据中的每一帧声学特征打上一个确定的 HMM 状态标签（例如，音素 ‘p-ah+b‘ 的第 2 个状态）。

DNN-HMM 的训练流程

这是一个多阶段的过程，需要一个已有的 GMM-HMM 系统来“引导” DNN 的训练。

1. 训练一个 **GMM-HMM** 系统:
   首先，使用我们前面介绍的 Baum-Welch (ML) 和判别式训练 (MMI/MPE) 方法，训练一个尽可能好的 GMM-HMM 系统。
2. 生成“强制对齐” **(Forced Alignment)**:
   使用上一步训练好的 GMM-HMM，对所有的训练语音和其对应的正确文本进行维特比解码。这会为训练数据中的每一帧声学特征打上一个确定的 HMM 状态标签（例如，音素 ‘p-ah+b‘ 的第 2 个状态）。
3. 训练 **DNN** 分类器:
   - 输入：当前帧的声学特征，并拼接其前后几帧（例如，使用一个包含 11 帧的窗口），以提供上下文信息。
   - 输出：一个 Softmax 层，其神经元数量等于 HMM 状态的总数。
   - 目标：最小化交叉熵损失。

这是一个多阶段的过程，需要一个已有的 GMM-HMM 系统来“引导”DNN 的训练。

1. 训练一个 **GMM-HMM** 系统：
   首先，使用我们前面介绍的 Baum-Welch (ML) 和判别式训练 (MMI/MPE) 方法，训练一个尽可能好的 GMM-HMM 系统。
2. 生成“强制对齐”**(Forced Alignment)**：
   使用上一步训练好的 GMM-HMM，对所有的训练语音和其对应的正确文本进行维特比解码。这会为训练数据中的每一帧声学特征打上一个确定的 HMM 状态标签（例如，音素 ‘p-ah+b‘ 的第 2 个状态）。
3. 训练 **DNN** 分类器：
   - 输入：当前帧的声学特征，并拼接其前后几帧（例如，使用一个包含 11 帧的窗口），以提供上下文信息。
   - 输出：一个 Softmax 层，其神经元数量等于 HMM 状态的总数。
   - 目标：最小化交叉熵损失。
4. 构建并评估混合系统：
   将训练好的 DNN 作为发射概率模型，与 GMM-HMM 的转移概率 A 组合，形成最终的 DNN-HMM 系统，用于解码测试集。

- 强大的建模能力:
  DNN 作为一种深度模型，可以学习到声学特征与 HMM 状态之间高度非线性的复杂关系，这是 GMM 无法比拟的。

DNN-HMM 的巨大优势

- 强大的建模能力:
  DNN 作为一种深度模型，可以学习到声学特征与 HMM 状态之间高度非线性的复杂关系，这是 GMM 无法比拟的。
- 利用丰富的上下文信息:
  通过在输入端拼接多帧特征（Context Window），DNN 在对当前帧进行分类时能够“看到”其上下文，做出更准确的判断。

DNN-HMM 的巨大优势

- 强大的建模能力:
  DNN 作为一种深度模型，可以学习到声学特征与 HMM 状态之间高度非线性的复杂关系，这是 GMM 无法比拟的。
- 利用丰富的上下文信息:
  通过在输入端拼接多帧特征（Context Window），DNN 在对当前帧进行分类时能够“看到”其上下文，做出更准确的判断。
- 简化特征工程:
  DNN 对特征的相关性不敏感，这意味着我们可以使用更简单、更接近原始信号的特征，如 **FBank (**滤波器组能量**)**，而不再局限于复杂的 MFCC。FBank 也被证明在 DNN 系统中比 MFCC 效果更好。

DNN-HMM 的巨大优势

- 强大的建模能力：
  DNN 作为一种深度模型，可以学习到声学特征与 HMM 状态之间高度非线性的复杂关系，这是 GMM 无法比拟的。
- 利用丰富的上下文信息：
  通过在输入端拼接多帧特征（Context Window），DNN 在对当前帧进行分类时能够“看到”其上下文，做出更准确的判断。
- 简化特征工程：
  DNN 对特征的相关性不敏感，这意味着我们可以使用更简单、更接近原始信号的特征，如 **FBank (**滤波器组能量**)**，而不再局限于复杂的 MFCC。FBank 也被证明在 DNN 系统中比 MFCC 效果更好。
- 性能的飞跃：
  在 2010 年代初，DNN-HMM 的出现为语音识别领域带来了革命性的突破，相比当时最先进的 GMM-HMM 系统，词错误率 (WER) 实现了约 **30%** 的相对下降，开启了深度学习在语音识别领域应用的新篇章。

- 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。
- 训练目标不一致：声学模型、语言模型、发音词典分开独立训练，并非端到端地联合优化。

我们已经走了多远？——混合系统的时代

通过用 DNN 替代 GMM，DNN-HMM 混合系统极大地提升了识别性能，开启了深度学习的革命。然而，这套系统依然是一个由多个独立部分拼接而成的复杂流水线。

我们已经走了多远？——混合系统的时代

通过用 DNN 替代 GMM，DNN-HMM 混合系统极大地提升了识别性能，开启了深度学习的革命。然而，这套系统依然是一个由多个独立部分拼接而成的复杂流水线。

一个大胆的设想：能否一步到位？

构建一个单一的、统一的神经网络，直接完成从声学特征到文本的映射。

$$\text{声学特征} \xrightarrow{\text{一个神奇的黑箱模型}} \text{文本}$$

通向端到端 (End-to-End) 之路

我们已经走了多远？——混合系统的时代

通过用 DNN 替代 GMM，DNN-HMM 混合系统极大地提升了识别性能，开启了深度学习的革命。然而，这套系统依然是一个由多个独立部分拼接而成的复杂流水线。

一个大胆的设想：能否一步到位？

构建一个单一的、统一的神经网络，直接完成从声学特征到文本的映射。

$$\text{声学特征} \xrightarrow{\text{一个神奇的黑箱模型}} \text{文本}$$

下次课预告：实现梦想的钥匙

为了实现这个目标，研究者们发明了多种革命性的架构，我们下次课将深入探索其中的三大主流技术：CTC, Attention, RNN-T.

推荐阅读

- 斯坦福大学 HMM 教程
- Hidden Markov Model Clearly Explained
- 台大李宏毅老师课程

致谢

本节 Slides 的制作参考了多位学者的优秀资料。部分内容与图示的出处（包括但不限于）如下：西北工业大学谢磊教授的课程 Slides、上海交通大学俞凯教授的课程 Slides、剑桥大学 Jingzhou Yang 的博士论文，以及网络视频“Hidden Markov Model Clearly Explained”等。在此谨致谢意。