

语音识别：从入门到精通

第五讲：基于GMM-HMM的语音识别系统

主讲人 张彬彬

西北工业大学

binbzha@gmail.com





背景知识回顾 (重要)

1. 特征提取

- a. 数字信号处理的基本知识
- b. MFCC/Fbank特征

2. 混合高斯模型GMM

- a. GMM模型
- b. EM算法

3. 隐马尔可夫模型HMM

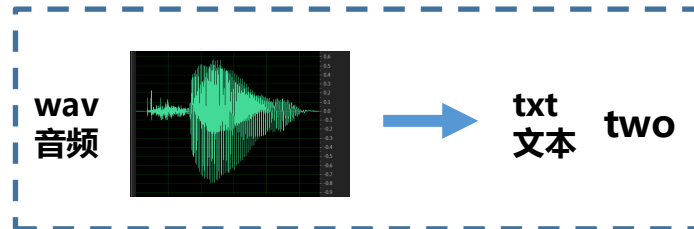
- a. HMM的三个基本问题 (概率问题、学习问题、预测问题)



GMM-HMM语音识别框架与概念

终极目的：让机器“听懂”。

- 对齐：“音频wav”和“文本txt”的对应关系
- 训练：已知对齐（wav及其txt），迭代计算模型参数。
- 解码：根据训练得到的模型参数，从wav推出txt。



- 数据源准备 (wav/txt)
- 其他数据准备 (词典、音素集、问题集等)
- 验证集，测试集

- MFCC

- 单音素为三音素提供对齐
- 多次三音素训练，逐层依赖



内容提要

1. 基于孤立词的GMM-HMM语音识别系统
 - a. **训练** (前向后向训练/Viterbi训练)
 - b. **解码**
2. 基于单音素的GMM-HMM语音识别系统
 - a. 音素/词典
 - b. **训练**
 - c. **解码**
3. 基于三音素的GMM-HMM语音识别系统
 - a. 三音素
 - b. 决策树
 - c. **训练**
 - d. **解码**
4. 基于GMM-HMM语音识别系统流程
5. 作业

语音识别系统的**训练和解码**是本节课程需要反复强化理解的内容

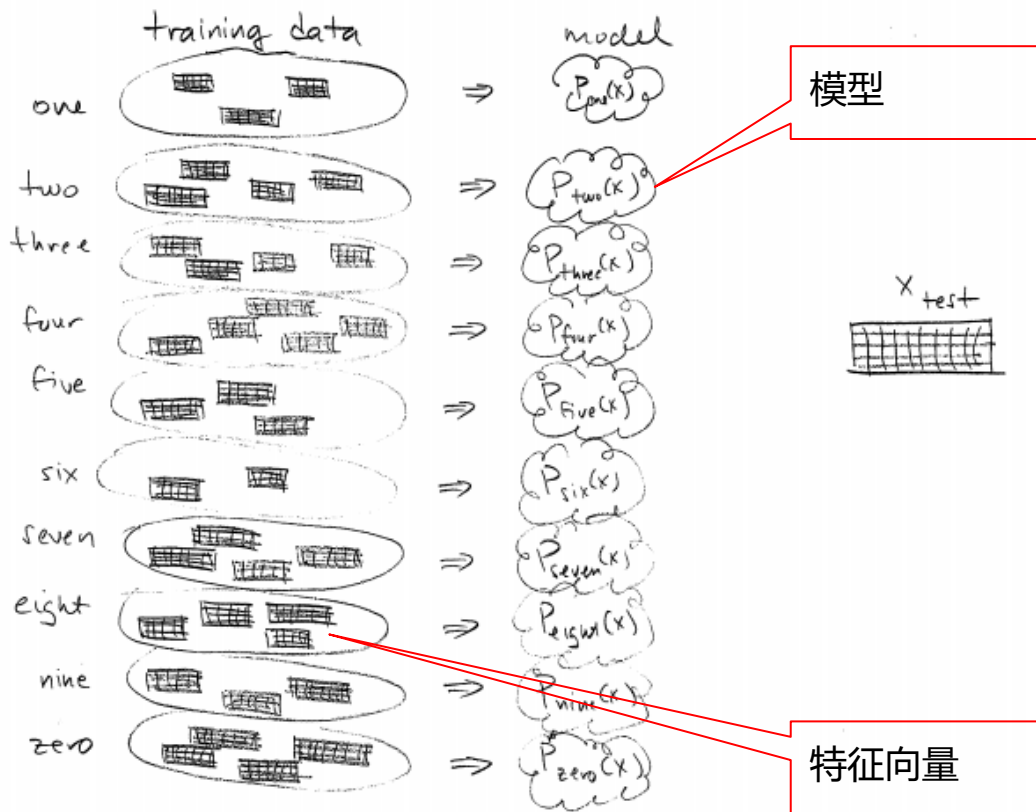


基于孤立词的GMM-HMM语音识别系统

考虑一个最简单的0~9十个数字的
孤立词语音识别系统？

- 数据
- 建模
- 如何训练
- 如何测试（解码）

是不是有点像语音的mnist?





目标

X_{test} 测试特征, $P_w(X)$ 是词 w 的概率模型, $vocab$ 是词表 (在该示例中即0~9 10个数字)

$$answer = \arg \max_{w \in vocab} P_w(X_{test})$$

- 假设我们为每个词建立了一个模型 $P_{one}(X), P_{two}(X), P_{three}(X) \dots$
- 计算在每个词上的概率
- 选择所有词中概率最大的词作为识别结果
- 问题:
 - 你想到那些建模方法可以表示 $P_w(X)$, DNN, GMM?
 - 语音任务的特点? 序列性, 不定长性
 - 如何解决这些问题?



词（语音）是一个序列， $P_w(X)$ 可以用HMM的概率问题来描述，并且其中的观测是连续概率密度分布。

回想一下GMM-HMM

- GMM概率密度建模
- HMM序列建模

现在我们也有了这10个词的训练数据，**我们可以做什么？**

Yes，为每个词建立一个GMM-HMM模型。



GMM-HMM语音识别系统流程（孤立词）

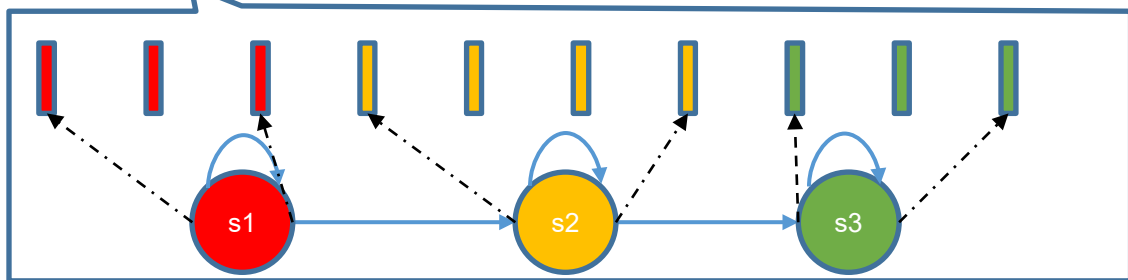
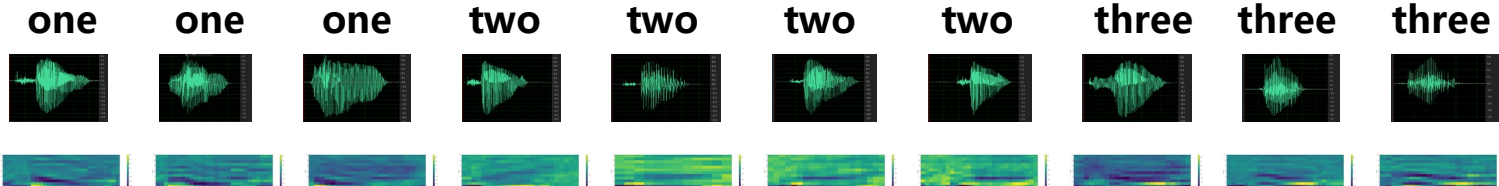
训练

数据准备
音素，词典，
训练音频/文本

特征提取
MFCC

HMM状态
序列建模

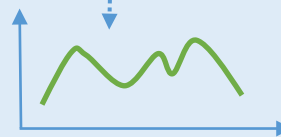
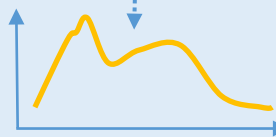
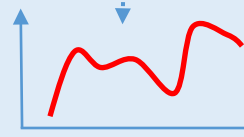
GMM模型
概率密度建模



更新参数 ($\alpha_{1jm}, \mu_{1jm}, \Sigma_{1jm}$)

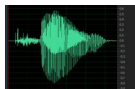
更新参数 ($\alpha_{2jm}, \mu_{2jm}, \Sigma_{2jm}$)

更新参数 ($\alpha_{3jm}, \mu_{3jm}, \Sigma_{3jm}$)

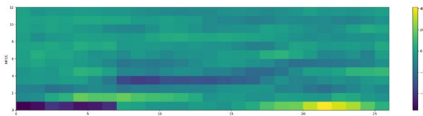


测试（解码）

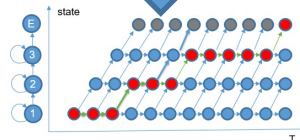
未知
wav



提取
特征



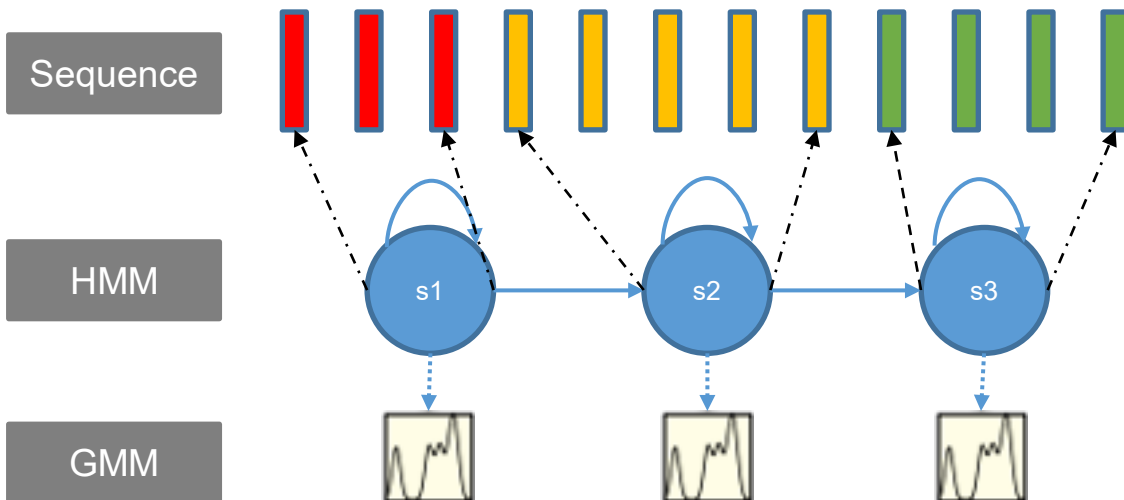
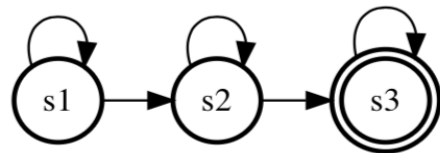
Viterbi
解码图



解码
结果 two



- 语音识别中的GMM (对角的GMM, 协方差为对角阵, MFCC特征)
- 语音识别中的HMM
 - 3状态, 为什么?
 - 左右模型的HMM(left-to-right HMM), 为什么?
 - 拓扑结构(s1, s2, s3为状态)
- 如何结合?





- 从系统的角度思考
 - 输入：词 w 和 w 所对应的训练数据
 - 输出：词 w 的HMM-GMM模型，也就是其参数
- 关键点
 - 任务：训练数据 $X_{w1}, X_{w2}, X_{w3} \dots$ 中训练 $P_w(X)$ ，估计HMM-GMM参数
 - 准则：最大似然
 - 方法：
 - Viterbi学习(Viterbi训练)
 - Baum-Welch学习(前向后向训练)



- 回顾一下都有哪些参数？
 - 初始参数（从左到右HMM）
 - 转移参数
 - 观测参数（对角GMM模型）
 - 混合系数
 - 均值
 - 方差

• 无隐变量模型最大似然估计

- count (hard)
- normalize (M步)

- 全班共4个学生，其中2个男生，身高分别为1.6和1.7，求全班男生的平均身高。

$$h = \frac{1.6 + 1.7}{2} = 1.65$$

$$= \frac{1 * 1.6 + 1 * 1.7 + 0 * h1 + 0 * h2}{1 + 1 + 0 + 0}$$

• 含隐变量模型最大似然估计

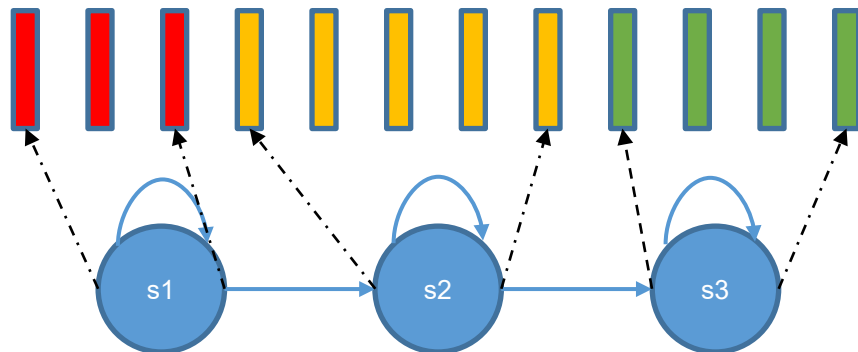
- count (soft, E步)
- normalize (M步)

- 全班共4个学生，不知道其确定性别，只知道其属于其是男生的概率分别为0.1, 0.3, 0.5, 0.8，身高分别为1.6, 1.7, 1.7, 1.8，求全班男生平均身高。

$$h = \frac{0.1 * 1.6 + 0.3 * 1.7 + 0.5 * 1.7 + 0.8 * 1.8}{0.1 + 0.3 + 0.5 + 0.8} = 1.74$$



- 类比问题：性别已知，求男生的平均身高。
- E步(hard count)
 - Viterbi算法得到最优的状态序列（**对齐 alignment**），在 t 时刻处于状态 i 上的概率(非0即1)
 - GMM模型中在 t 时刻处于状态 i 第 k 个GMM分量的概率（依然是soft count）
- M步(normalize)
 - 更新转移参数、GMM参数（混合系数、均值、方差）
- 重复E/M





学习算法：Viterbi学习算法

• Viterbi学习算法总结：

1. 初始化GMM-HMM参数 $\lambda = (\pi_i, a_{ij}, \text{MM 参数})$, 其中每个状态 j 对应的GMM的参数为 $(\alpha_{jm}, \mu_{jm}, \Sigma_{jm})$
2. 基于GMM-HMM参数 λ 和Viterbi算法得到状态-观测对齐, 得到每个观测对应的隐藏状态
3. 更新参数 λ
 - $\hat{\pi}_i = \frac{C(i)}{\sum_k C(k)}$, $C(i)$ 表示初始状态为 i 的次数
 - $\hat{a}_{ij} = \frac{C(i \rightarrow j)}{\sum_k C(i \rightarrow k)}$, $C(i \rightarrow j)$ 表示从状态 i 到状态 j 的转移次数
 - 用算法9.2更新GMM的参数 $(c_{jm}, \mu_{jm}, \Sigma_{jm})$
4. 重复2, 3步, 直到收敛

算法 9.2 (高斯混合模型参数估计的 EM 算法)

输入：观测数据 y_1, y_2, \dots, y_N , 高斯混合模型；

输出：高斯混合模型参数。

(1) 取参数的初始值开始迭代

(2) E 步：依据当前模型参数, 计算分模型 k 对观测数据 y_j 的响应度

$$\hat{\gamma}_{jk} = \frac{\alpha_k \phi(y_j | \theta_k)}{\sum_{k=1}^K \alpha_k \phi(y_j | \theta_k)}, \quad j=1, 2, \dots, N; \quad k=1, 2, \dots, K$$

(3) M 步：计算新一轮迭代的模型参数

$$\begin{aligned} \hat{\mu}_k &= \frac{\sum_{j=1}^N \hat{\gamma}_{jk} y_j}{\sum_{j=1}^N \hat{\gamma}_{jk}}, \quad k=1, 2, \dots, K \\ \hat{\sigma}_k^2 &= \frac{\sum_{j=1}^N \hat{\gamma}_{jk} (y_j - \mu_k)^2}{\sum_{j=1}^N \hat{\gamma}_{jk}}, \quad k=1, 2, \dots, K \\ \hat{\alpha}_k &= \frac{\sum_{j=1}^N \hat{\gamma}_{jk}}{N}, \quad k=1, 2, \dots, K \end{aligned}$$

(4) 重复第 (2) 步和第 (3) 步, 直到收敛。



前向后向训练 (Baum-Welch训练)

- 类比问题：知道每人属于男生的性别概率，求男生的平均身高。
- E步(soft count)
 - 前向算法+后向算法，在时刻 t 处于状态 i 的概率
 - 在时刻 t 处于状态 j 且为GMM第 k 个分量的概率
- M步(normalize)
 - 更新转移参数、GMM参数（混合系数、均值、方差）
- 重复E/M



学习算法：Baum-Welch学习算法

• Baum-Welch学习算法总结

1. 初始化GMM-HMM参数 $\lambda = (\pi_{\pm}, a_{ij}, (c_{jm}, \mu_{jm}, \Sigma_{jm}))$

2. E步：对所有时间 t 、状态 i

- 递推计算前向概率 $\alpha_t(i)$ 和后向概率 $\beta_t(i)$

- 计算 $\zeta_t(j, k) = \frac{\sum_i \alpha_{t-1}(i) a_{ij} c_{jk} b_{jk}(o_t) \beta_t(j)}{\sum_{i=1}^N \alpha_T(i)}$, $\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \alpha_T(i)}$, $\gamma_t(i) = \sum_{k=1}^N \xi_t(i, k)$

3. M步：更新参数

$$\hat{\mu}_{jk} = \frac{\sum_{t=1}^T \zeta_t(j, k) o_t}{\sum_{t=1}^T \zeta_t(j, k)}$$

$$\hat{\Sigma}_{jk} = \frac{\sum_{t=1}^T \zeta_t(j, k) (o_t - \hat{\mu}_{jk})(o_t - \hat{\mu}_{jk})^T}{\sum_{t=1}^T \zeta_t(j, k)}$$

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

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

$$\hat{\pi}_{\pm} = \gamma_{\pm}(i)$$

4. 重复2, 3步，直到收敛



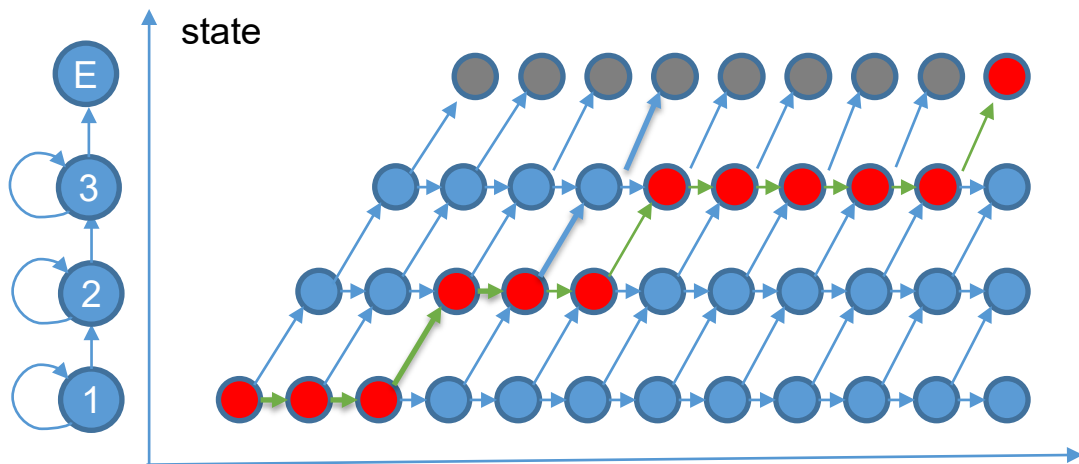
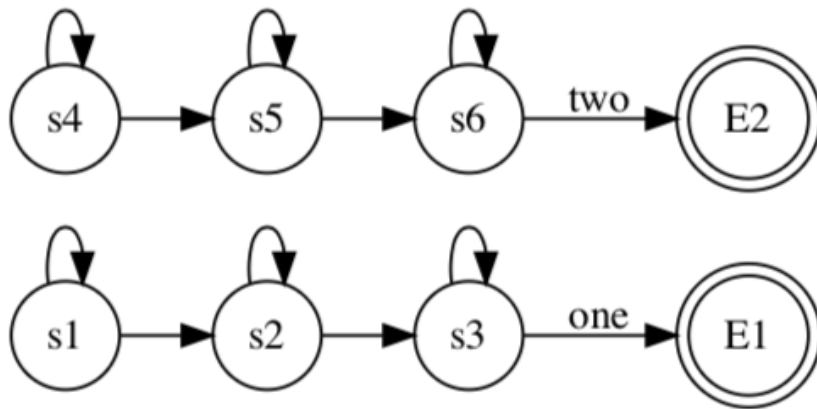
解码

- 从系统的角度思考
 - 输入
 - 各个词的HMM-GMM模型
 - 未知的测试语音 X_{test}
 - 输出:
 - X_{test} 是哪个词
- 关键点
 - 概率问题对所有的 w , 如何 计算 $P_w(X_{test})$
 - 方法:
 - 前向算法
 - Viterbi算法 (可以回溯到最优的状态序列)



解码

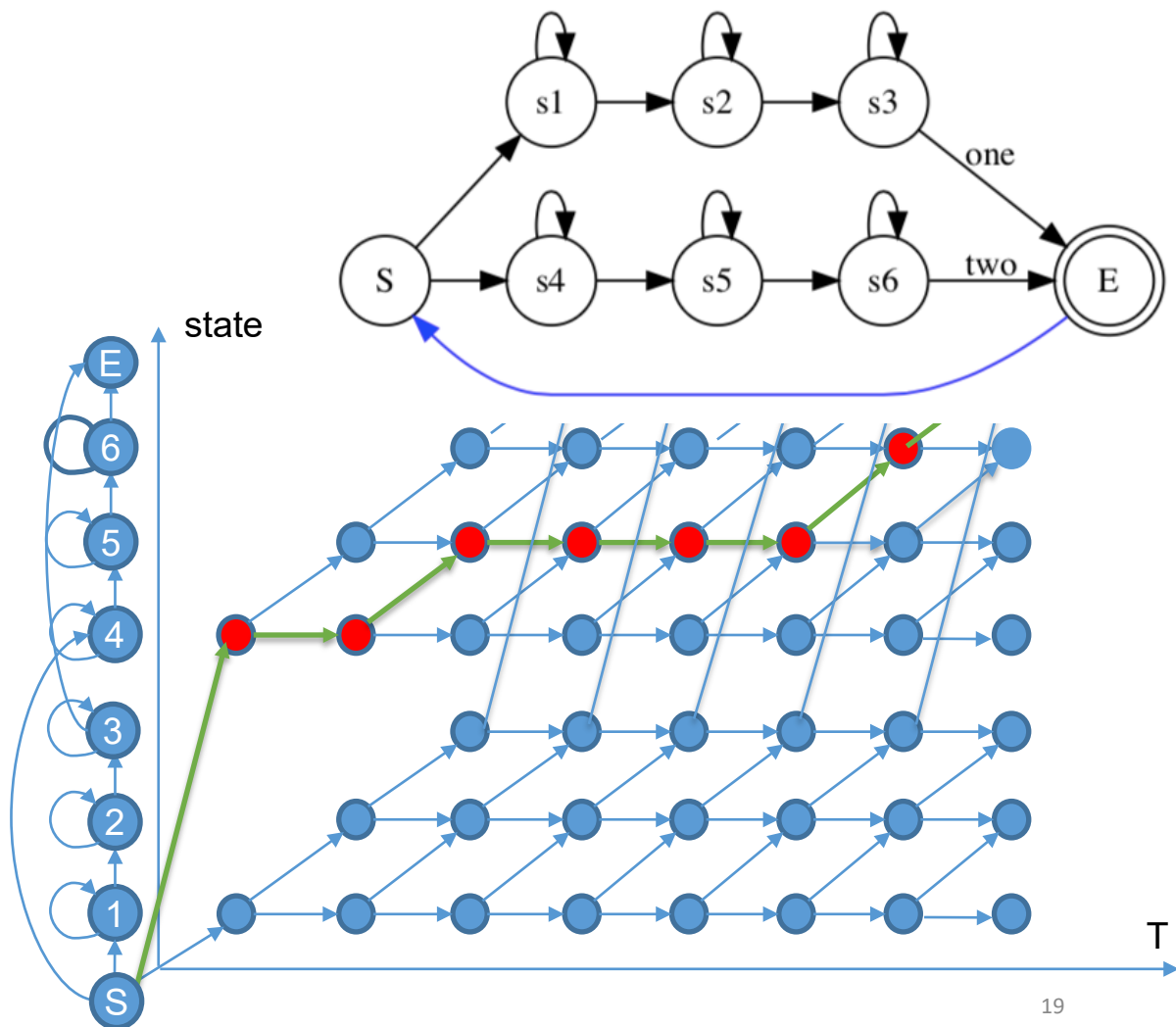
- one two两个数字识别问题
- 分散的解码图
 - 前向算法
 - Viterbi算法





解码

- one two两个数字识别问题
- 紧凑的解码图
 - Viterbi算法
- 思考：如何构造连续数字one-two识别的解码图？



- 孤立词系统的缺点：
 - 建模单元数、计算量和词典大小成正比
 - 词的状态数 (a/accomplishment)对每个词应该不同，长词应该使用更多的状态。
 - OOV(Out of Vocabulary) 的问题
 - 实际上，词并不是一个语言的基本发音单元，以词为建模单元无法共享这些发音的基本单元。如：
 - cat hat dad bad ... 中的a均发/a/的音
 - 包 操 刀 高 交 中的韵母均发/ao/的音



音素(Phone)

- 发音的基本单元: 音素
- 静音Silence(SIL)

英文音素 (CMU phone, 39)

AA AE AH AO AW AX AXR AY

B BD CH D DD DH DX EH ER EY

F G GD HH IH IX IY JH K

KD L M N NG OW OY P PD

R S SH T TD TH TS UH UW

V W X Y Z ZH

中文音素 (可以认为声韵母就是音素)

a o e i u v

b p m f d t n l g k h j q x

zh ch sh z c s y w

ai ei ui ao ou iu ie ue er

an en in un vn

ang eng ing ong



词典 (Lexicon)

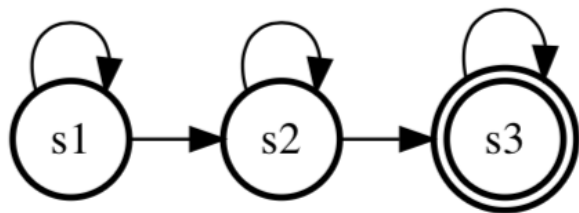
- 词到音素序列的映射(文件), 0 ~ 9 10个数字的词典如下

one	W AA N
two	T UW
three	TH R IY
four	F AO R
five	F AY V
six	S IH K S
seven	S EH V AX N
eight	EY T
nine	N AY N
zero	Z IY R OW



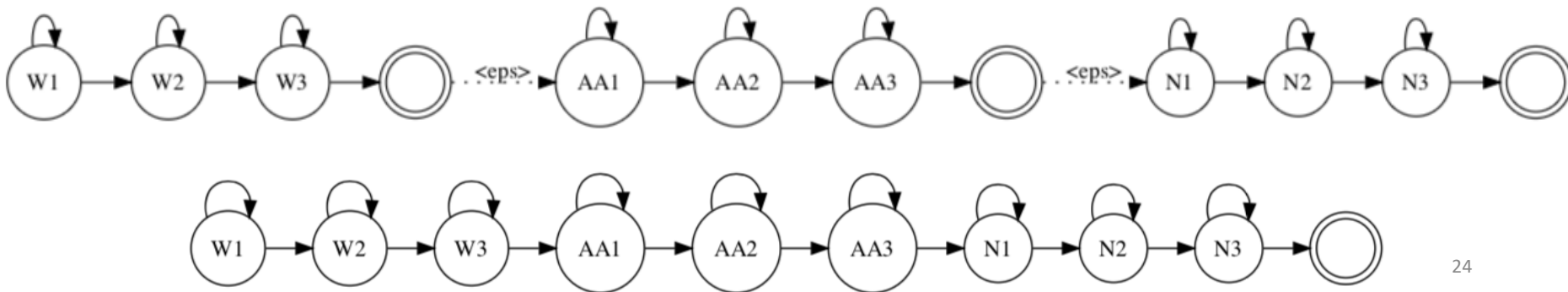
单音素HMM拓扑结构

- 每个音素使用经典的3状态结构



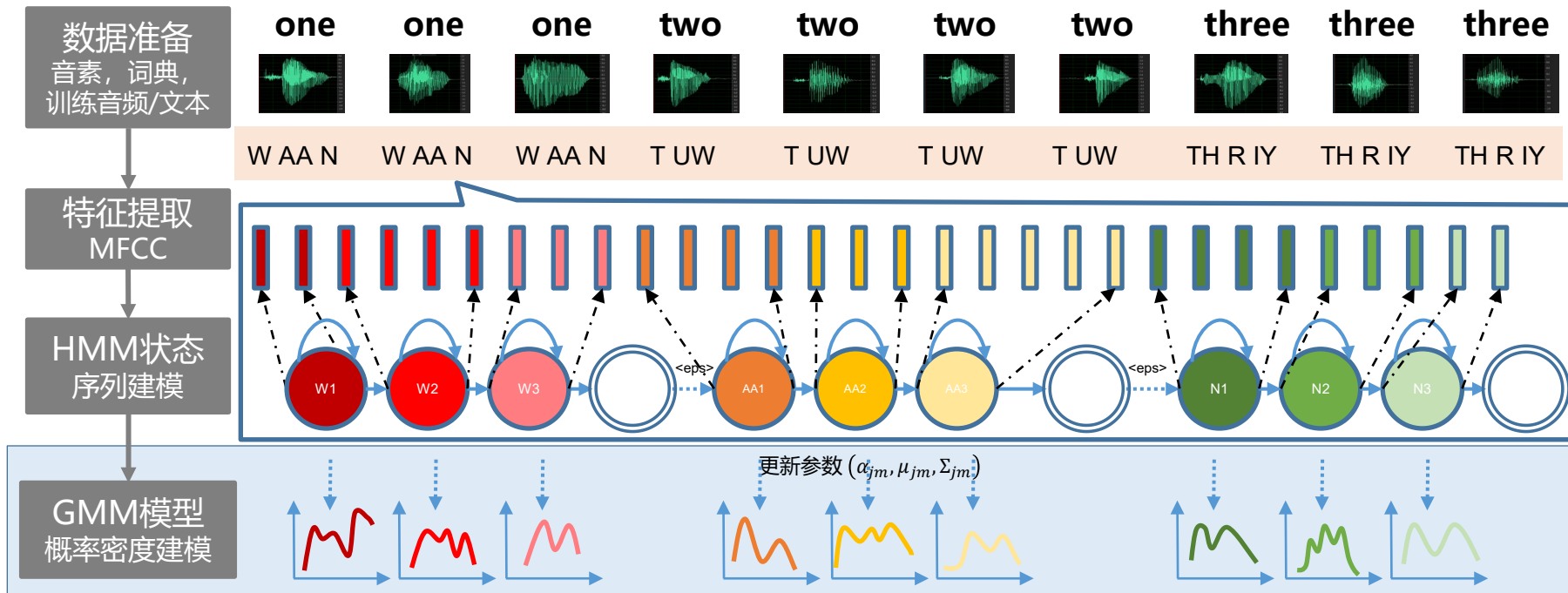


- 现在假设一句话里面包含一个单词，例如one(W AA N)
 - 如何做Viterbi训练
 - 如何做前向后向训练
- 问题1：如果一句话中包含多个单词，如何做训练？
- 问题2：假设单词中有多音字，怎么办？





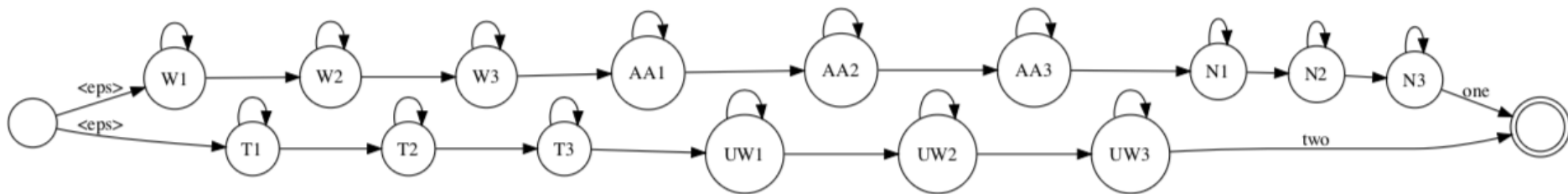
GMM-HMM语音识别系统流程（单音素：训练）





解码? (思考一下)

- 基于单音素的解码图



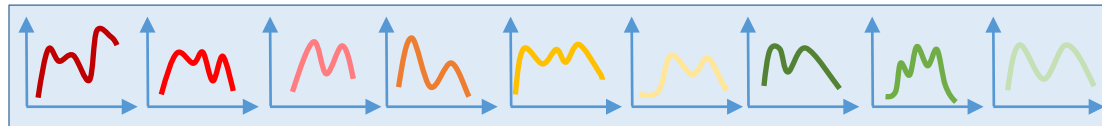
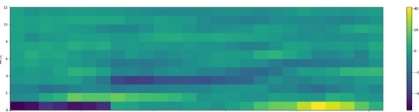
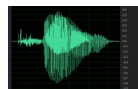


GMM-HMM语音识别系统流程（单音素：解码）

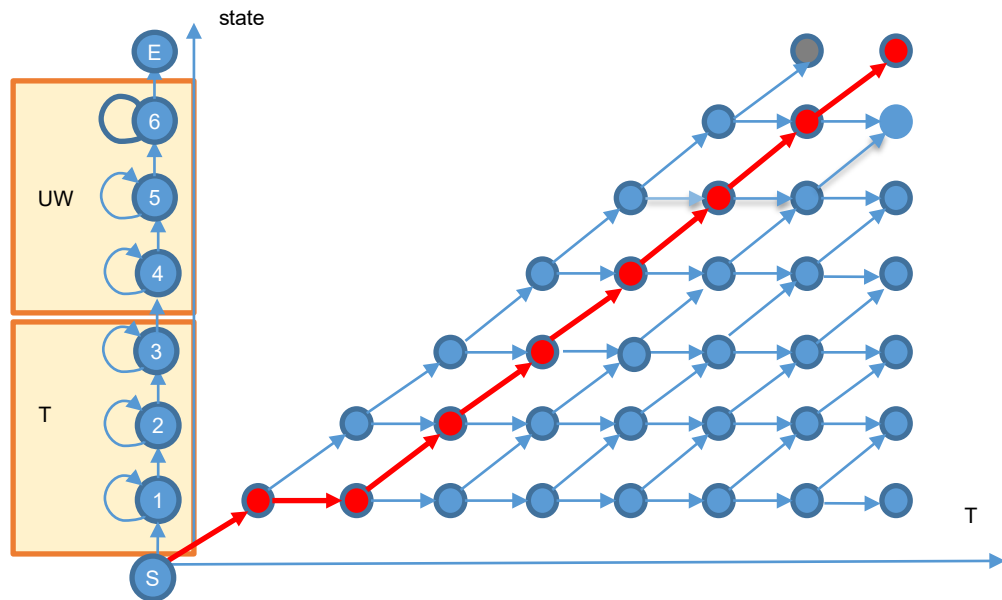
未知wav

提取特征

GMM模型



Viterbi算法



解码
结果

T UW



two

基于三音素的GMM-HMM语音识别系统

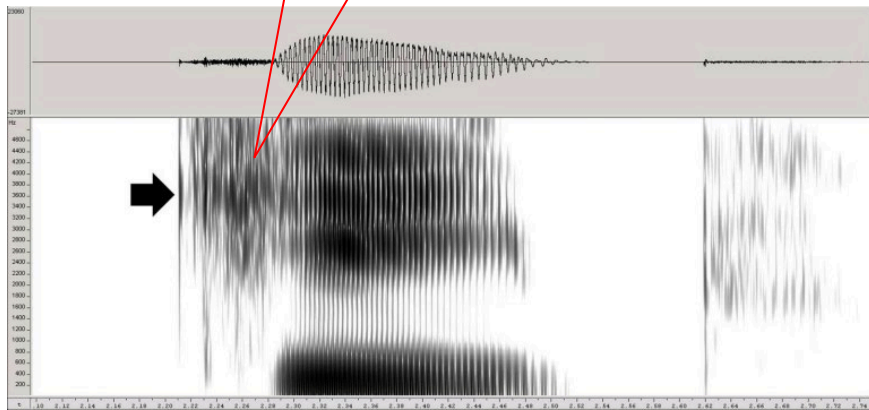
单音素缺点1: 建模单元数少

- 一般英文系统的音素数量在30 ~ 60个
- 一般中文系统的音素数量在100个左右

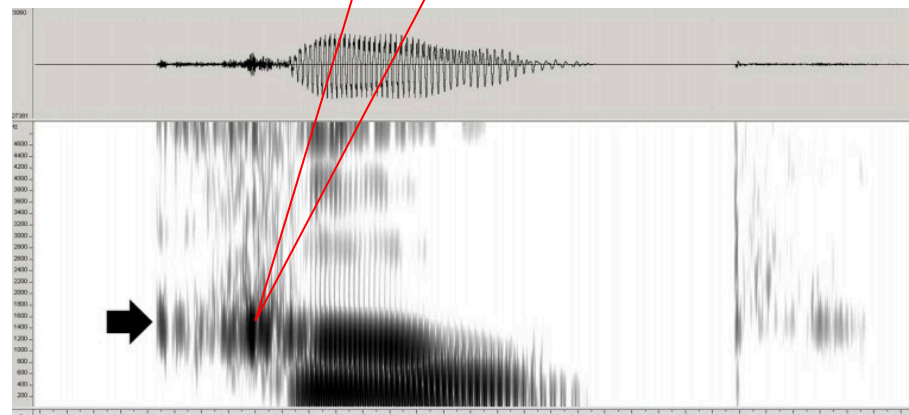
单音素缺点2: 音素的发音受其所在上下文的影响 (协同发音)

- 连读: Not at all, He is
- 吞音: first time

keep /K IY P/



coop /K UW P/





- 解决方案：考虑音素的上下文（Context），一般的，考虑前一个/后一个，称之为**三音素**，表示为A-B+C

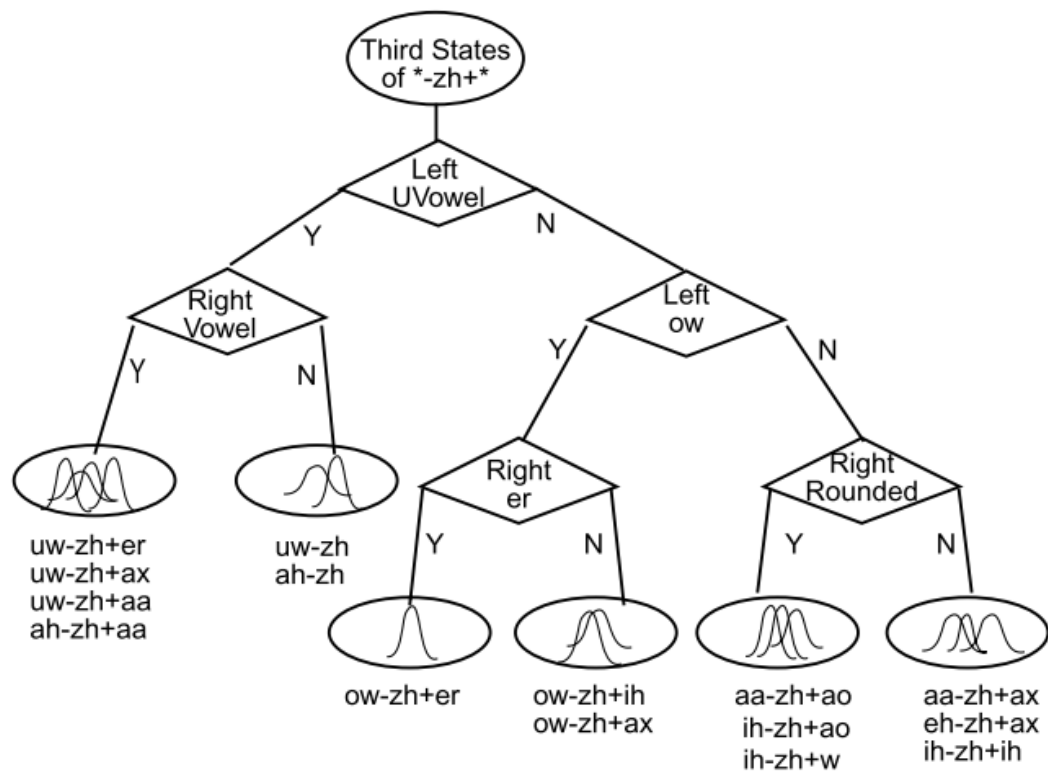
例如：KEEP K IY P => #-K+YI, K-IY+P, YI-P+#

- 问题1: 假设有N个音素，一共有多少个三音素？
- 问题2: 有的三音素训练数据少或不存在，怎么办？ B-B+B, Z-Z+Z
- 问题3: 有的三音素在训练中不存在，但在测试中有怎么办？
- 好像，三音素带来了新的问题？



绑定(Tying)

- 基本思想：上下文发音相近的三音素共享参数
- 自底向上：聚类
 - 没见过的数据可以解决吗？
- 自顶向下：决策树
 - 三音素绑定的实际解决方案



决策树长这样：

- 二叉树
- 每个非叶子节点上都会有一个问题
- 叶子节点是一个绑定三音素的集合
- 绑定的粒度为状态
 - A-B+C和A-B+D的第1个状态绑定在一起，并不代表其第2/3个状态也要在一起
 - 也就是B的每个状态都有一颗小的决策树
- zh-zh+zh



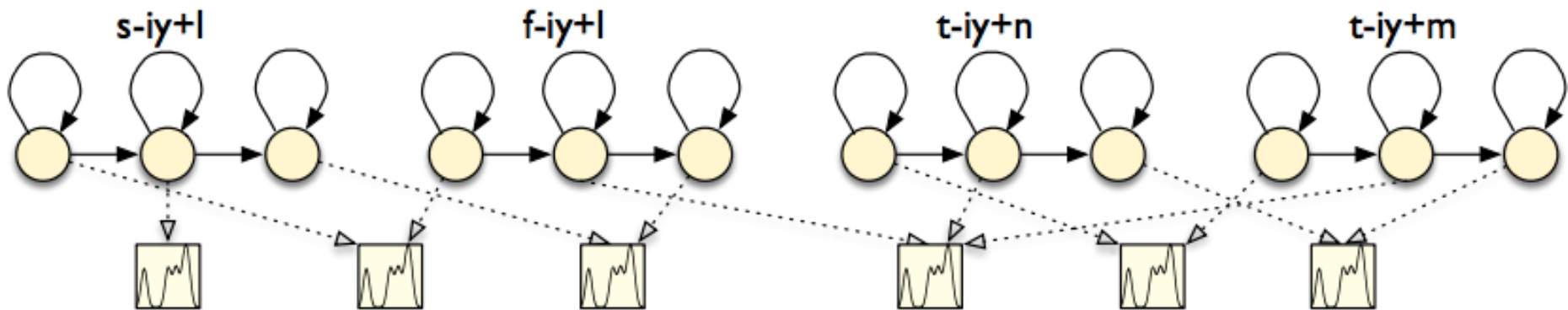
问题 (集)

- 常见问题
 - 元音(Vowel): AA AE AH AO AW AX AXR AY EH ER ...
 - 爆破音(Stop): B D G P T K
 - 鼻音(Nasal): M N NG
 - 摩擦音(Fricative): CH DH F JH S SH TH V Z ZH
 - 流音(Liquid): L R W Y
- 位置: 左/右
- 问题集的构建
 - 语言学家定义
 - Kaldi中通过自顶向下的聚类自动构建问题集



基于状态的绑定

- Context dependent State(CD-State)
- Senone



State-clustered triphones (GMMs)



决策树的构建（最优问题）

- 选择哪个问题进行二分类？
- 数据 $S = (x_1, \dots, x_m) \in (\mathbb{R}^N)^m$
- 模型：假设其服从单高斯分布，并且各维独立，也就是对角GMM

$$\Pr[x] = \frac{1}{\prod_{k=1}^N (2\pi\sigma_k^2)^{1/2}} \prod_{k=1}^N \exp\left(-\frac{1}{2} \frac{(x_k - \mu_k)^2}{\sigma_k^2}\right)$$

- 似然 Likelihood

$$\begin{aligned} L(S) &= -\frac{1}{2} \sum_{i=1}^m \left[\sum_{k=1}^N \log(2\pi\sigma_k^2) + \sum_{k=1}^N \frac{(x_{ik} - \mu_k)^2}{\sigma_k^2} \right] \\ &= -\frac{1}{2} \left[m \sum_{k=1}^N \log(2\pi\sigma_k^2) + m \sum_{k=1}^N \frac{\sigma_k^2}{\sigma_k^2} \right] \\ &= -\frac{1}{2} \left[mN(1 + \log(2\pi)) + m \sum_{k=1}^N \log(\sigma_k^2) \right]. \end{aligned}$$



决策树的构建（最优问题）

- 分成两类：

$$L(S_l) + L(S_r) = -\frac{1}{2}mN(1 + \log(2\pi)) - \frac{1}{2}\left[m_l \sum_{k=1}^N \log(\sigma_{lk}^2) + m_r \sum_{k=1}^N \log(\sigma_{rk}^2)\right]$$

- 似然增益(Likelihood gain)

$$L(S_l) + L(S_r) - L(S)$$

- 最优问题 q^*

$$q^* = \operatorname{argmin}_q \left[m_l \sum_{k=1}^N \log(\sigma_{lk}^2) + m_r \sum_{k=1}^N \log(\sigma_{rk}^2) \right]$$

$$\sigma_{lk}^2 = \frac{1}{m_l} \sum_{x \in S_l} x_k^2 - \frac{1}{m_l^2} \left(\sum_{x \in S_l} x_k \right)^2$$

$$\sigma_{rk}^2 = \frac{1}{m_r} \sum_{x \in S_r} x_k^2 - \frac{1}{m_r^2} \left(\sum_{x \in S_r} x_k \right)^2.$$



决策树的构建

1. 初始状态（一个结点）
2. 选择一个结点
 - 从问题集中选择似然增益最大的问题作为该节点问题
 - 建立该节点左右子节点，并将该节点上的统计量分为两部分
3. 重复2，直至
 - 达到一定数量的叶子结点
 - 似然增益小于某个阈值

决策树细节见文档《语音识别系列之决策树》



基于GMM-HMM语音识别系统流程

- 数据准备：音素列表、词典、训练数据（音频/文本）
- 特征提取：MFCC特征
- 单音素GMM-HMM：Viterbi训练
- 三音素GMM-HMM：决策树和三音素，Viterbi训练
- 解码
- 思考：为什么先做单音素训练？

数据准备



特征提取



单音素
GMM-HMM



三音素
GMM-HMM



解码



本章总结

1. 基于孤立词的GMM-HMM语音识别系统
 - a. 训练 (前向后向训练/Viterbi训练)
 - b. 解码
2. 基于单音素的GMM-HMM语音识别系统
 - a. 音素/词典
 - b. 训练
 - c. 解码
3. 基于三音素的GMM-HMM语音识别系统
 - a. 三音素
 - b. 决策树
 - c. 训练
 - d. 解码
4. 基于GMM-HMM语音识别系统流程



作业

- 作业来源：哥伦比亚大学语音识别课程[E6870](#)
- 难度系数：9
- 程序设计语言：C++
- 预计花费时间：5~10个小时
- **为什么选择该作业？**
 - 该作业质量很高（详细的作业说明，优秀的设计框架）
 - 如果能系统独立正确的完成本次作业，说明你对前5章学习的内容真正的理解掌握了，那很赞，👍
 - 学有余力的同学，可以深入研究本次作业的框架、细节实现等，对照本次课程梳理前向后向训练、Viterbi训练，Viterbi解码的过程。



作业

- 作业地址: https://github.com/nwpuaslp/ASR_Course/tree/master/05-GMM-HMM/
- 作业内容:
 - Viterbi解码
 - 估计GMM参数
 - 前向后向训练, 利用前向后向训练估计GMM参数
- 作业中的几个重要文件:
 - README.md: 如何安装、编译、填写代码、对比结果。
 - lab2.pdf: 原始作业说明, 需要细读。
 - src: src目录下为源代码文件。
 - lab2.txt: 提示思考的几个问题。



作业

在这里填上你的代码

```
28     throw runtime_error("GMM doesn't have single component.");
29     int gaussIdx = m_gmmSet.get_gaussian_index(gmmIdx, 0);
30     int dimCnt = m_gmmSet.get_dim_count();
31
32     // BEGIN_LAB
33     //
34     // Input:
35     //     "dimCnt" holds the dimension of the Gaussian and the
36     //     acoustic feature vector.
37     //     The acoustic feature vector is held in
38     //     "feats[0 .. (dimCnt-1)]".
39     //     "gaussIdx" is the index of the Gaussian to be updated.
40     //     "posterior" is the posterior count of this Gaussian for
41     //     the current frame.
42     //
43     //     The values of the current means and variances can be
44     //     accessed via the object "m_gmmSet".
45     //
46     // Output:
47     //     You should update the counts stored in
48     //
49     //     m_gaussCounts[0 .. (#gaussians-1)]
50     //     m_gaussStats1[0 .. (#gaussians-1), 0 .. (dimCnt - 1)]
51     //     m_gaussStats2[0 .. (#gaussians-1), 0 .. (dimCnt - 1)]
52     //
53     //     "m_gaussCounts" is intended to hold the total occupancy count
54     //     of each Gaussian; "m_gaussStats1" is intended for
55     //     storing some sort of first-order statistic for each
56     //     dimension of each Gaussian; and "m_gaussStats2" is intended for
57     //     storing some sort of second-order statistic for each
58     //     dimension of each Gaussian. The statistics you take
59     //     need to be sufficient for doing the reestimation step below.
60     //
61     //     These counts have all been initialized to zero
62     //     somewhere else at the appropriate time.
63
64     // suppose each GMM only has one component
65
66     // END_LAB
67     //
```



感谢聆听！