深蓝学院
shenlanxueyuan.com

音频语音与语言处理研究组
SLP
Audio, Speech and Language Processing Group NPU

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. Fbank/MFCC特征(分帧->预加重->加窗->FFT->Mel滤波器组->(DCT))
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）
    b. GMM-HMM

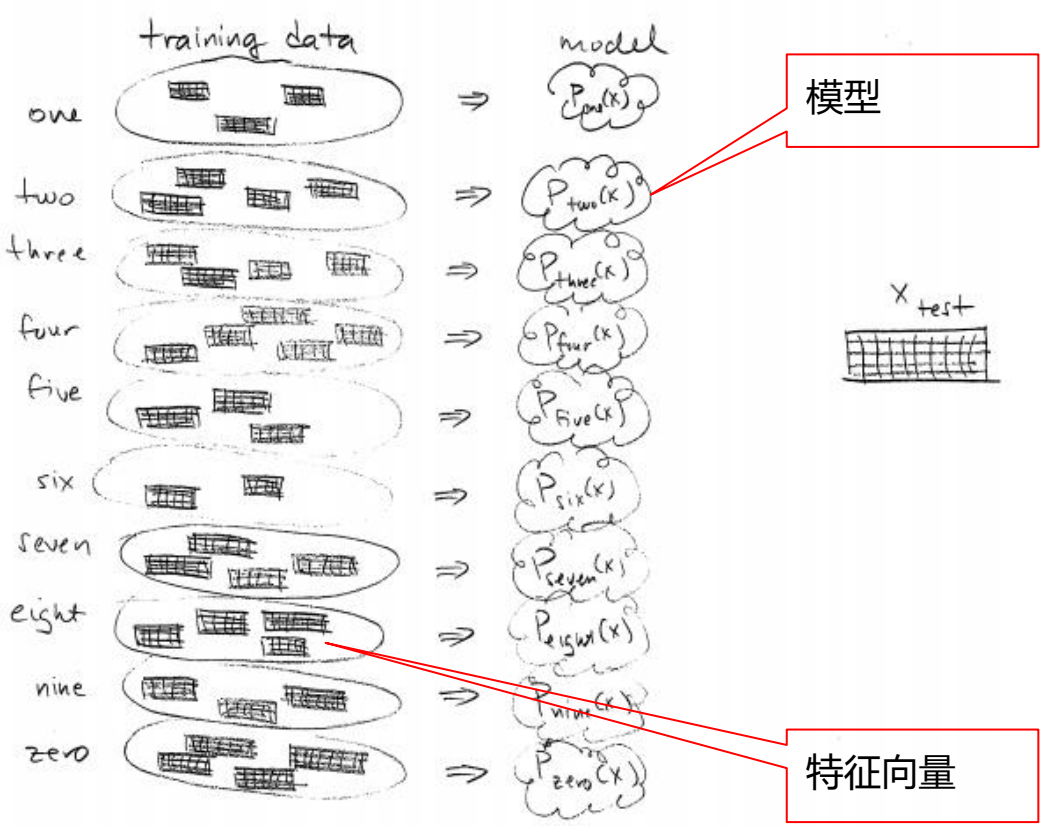
内容提要

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. Kaldi简介
6. 作业

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?

- 建模
- 如何训练
- 如何解码



目标

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

$$(\text{answer}) = \underset{\omega \in \text{vocab}}{\arg\max}\, P_{\omega}(\mathbf{x}_{\text{test}})$$

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

建模

词（语音）是一个序列模型，现在我们也有了这10个词的训练数据，我们可以做什么？

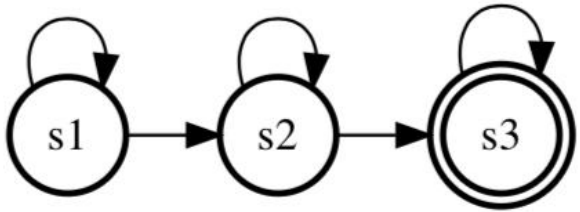
回想一下GMM-HMM

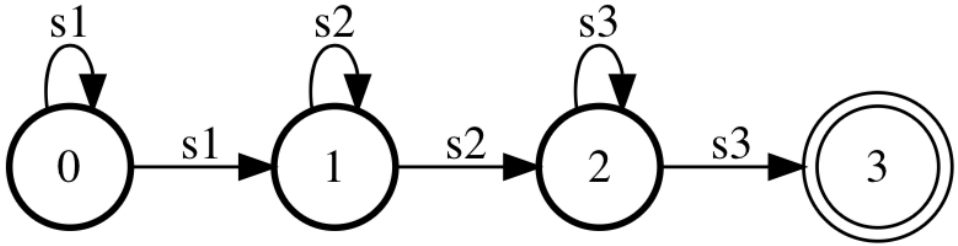
- GMM概率密度建模
- HMM序列建模

Yes，为每个词建立一个GMM-HMM模型。

建模

- 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 right HMM)，为什么？
  - 拓扑结构(s1, s2, s3为状态)
    - HMM状态在节点上（下左图，逻辑结构）
    - HMM状态在边上（下右图，WFST格式的表示，在实际中使用的更多）





训练

- 如何从训练数据$\boldsymbol{X}_{w1}, \boldsymbol{X}_{w2}, \boldsymbol{X}_{w3}\ldots$ 中训练$P_w(\boldsymbol{X})$
- 目标：估计HMM-GMM参数
- 准则：最大似然
- 方法：
  - Baum-Welch学习(前向后向训练)
  - Viterbi学习(Viterbi训练)

GMM-HMM参数

- 回顾一下都有那些参数？
  - ~~初始参数~~（左右HMM）
  - 转移参数
  - 观测参数（GMM模型）
    - 混合系数
    - 均值
    - 方差

训练

- 无隐变量模型最大似然估计
  - count （hard）
  - normalize （M步）
- 含隐变量模型最大似然估计
  - count（soft, E步）
  - normalize （M步）

前向后向训练（Baum-Welch训练）

- E步(count)
  - 前向算法+后向算法
  - 在时刻 t 处于状态 i且在时刻 $t+1$ 处于状态 j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 $\lambda = \left(\sout{\pi_i}, a_{ij}, \left(c_{jm}, \mu_{jm}, \Sigma_{jm}\right)\right)$
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

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

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

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

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

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

4. 重复2，3步，直到收敛

Viterbi训练

- 和K-means一样，我们可以做hard（硬）决策
- E步(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）
  - 在t时刻处于状态i第k个GMM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：Viterbi学习算法

- **Viterbi学习算法总结：**

1. 初始化GMM-HMM参数 $\lambda = (\pi_i, a_{ij}, \text{GMM参数})$，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~~
   - $\hat{a}_{ij} = \frac{C(i \to j)}{\sum_k C(i \to k)}$，$C(i \to j)$ 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 $y_1, y_2, \cdots, y_N$，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

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

（3）M 步：计算新一轮迭代的模型参数

$$\hat{\mu}_k = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk} y_j}{\sum_{j=1}^{N} \hat{\gamma}_{jk}}, \quad k = 1, 2, \cdots, K$$

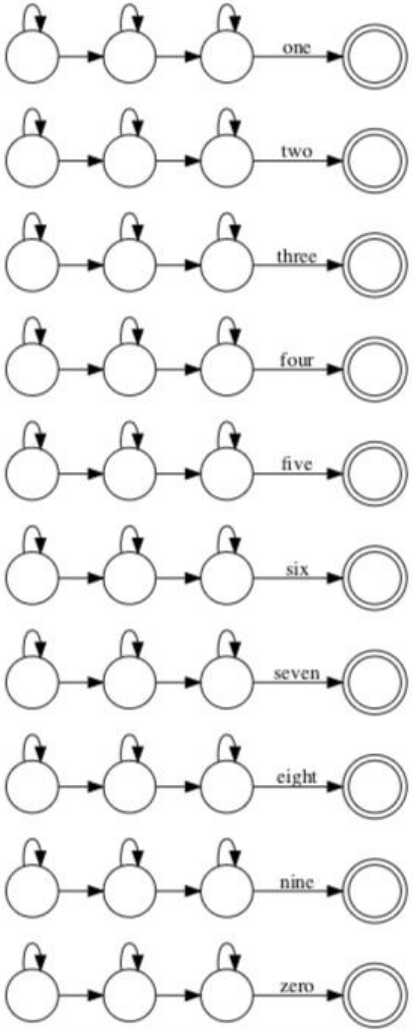
$$\hat{\sigma}_k^2 = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk} (y_j - \mu_k)^2}{\sum_{j=1}^{N} \hat{\gamma}_{jk}}, \quad k = 1, 2, \cdots, K$$

$$\hat{\alpha}_k = \frac{\sum_{j=1}^{N} \hat{\gamma}_{jk}}{N}, \quad k = 1, 2, \cdots, K$$

（4）重复第（2）步和第（3）步，直到收敛.

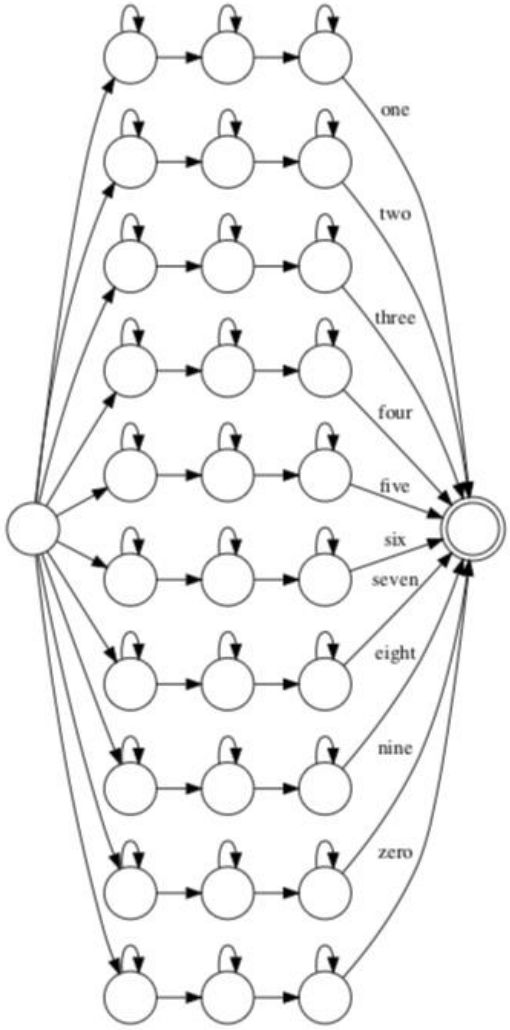
解码

- 解码图表示1
- 解码即识别，对所有的w，如何 计算$P_w(\boldsymbol{X}_{test})$
  - 前向算法
  - Viterbi算法
    - 可以回溯到最优的状态（词）序列



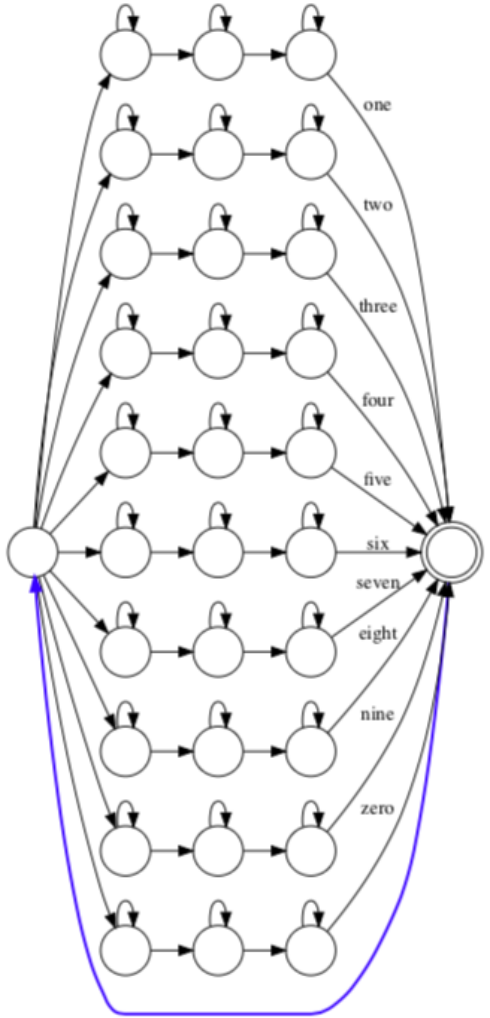
解码

- 解码图表示2
  - 更加紧凑
  - 灵活（孤立词、简单文法、语言模型等）
- 解码即识别，对所有的w，如何 计算$P_w(\boldsymbol{X}_{test})$？
  - ~~前向算法~~
  - Viterbi算法
    - 可以回溯到最优的状态（词）序列
    - 全局解码信息剪枝
      - rank
      - beam
- 问题：如何构造一个数字串识别的解码图？



解码

- 解码图表示2
  - 更加紧凑
  - 灵活
- 如何构造一个数字串识别的解码图？
  - 如图，连接这条蓝色的线即可



基于单音素的GMM-HMM语音识别系统

- 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

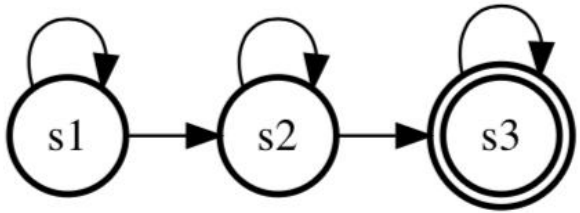
词典（Lexicon）

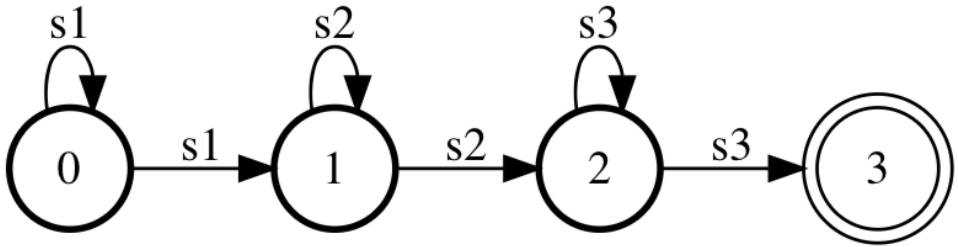
- 词到音素序列的映射(文件)， 0～9 10个数字的词典如下

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

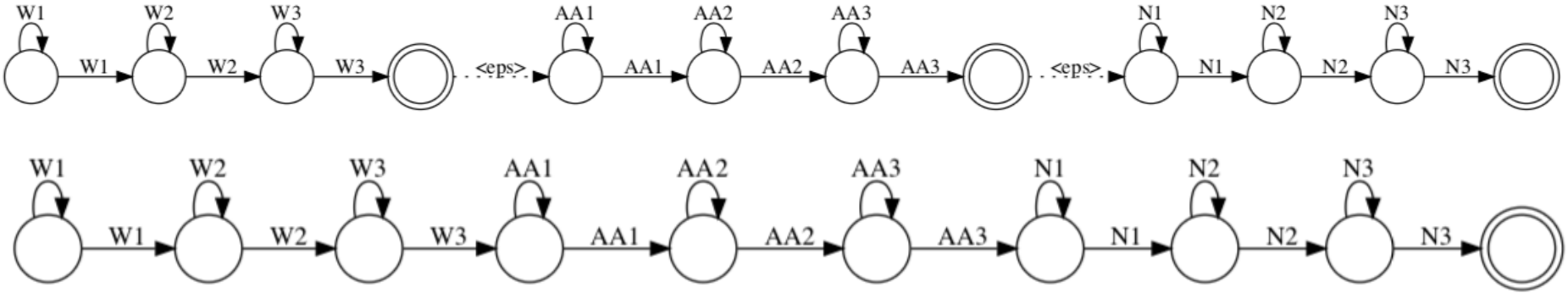
- 每个音素使用经典的3状态结构



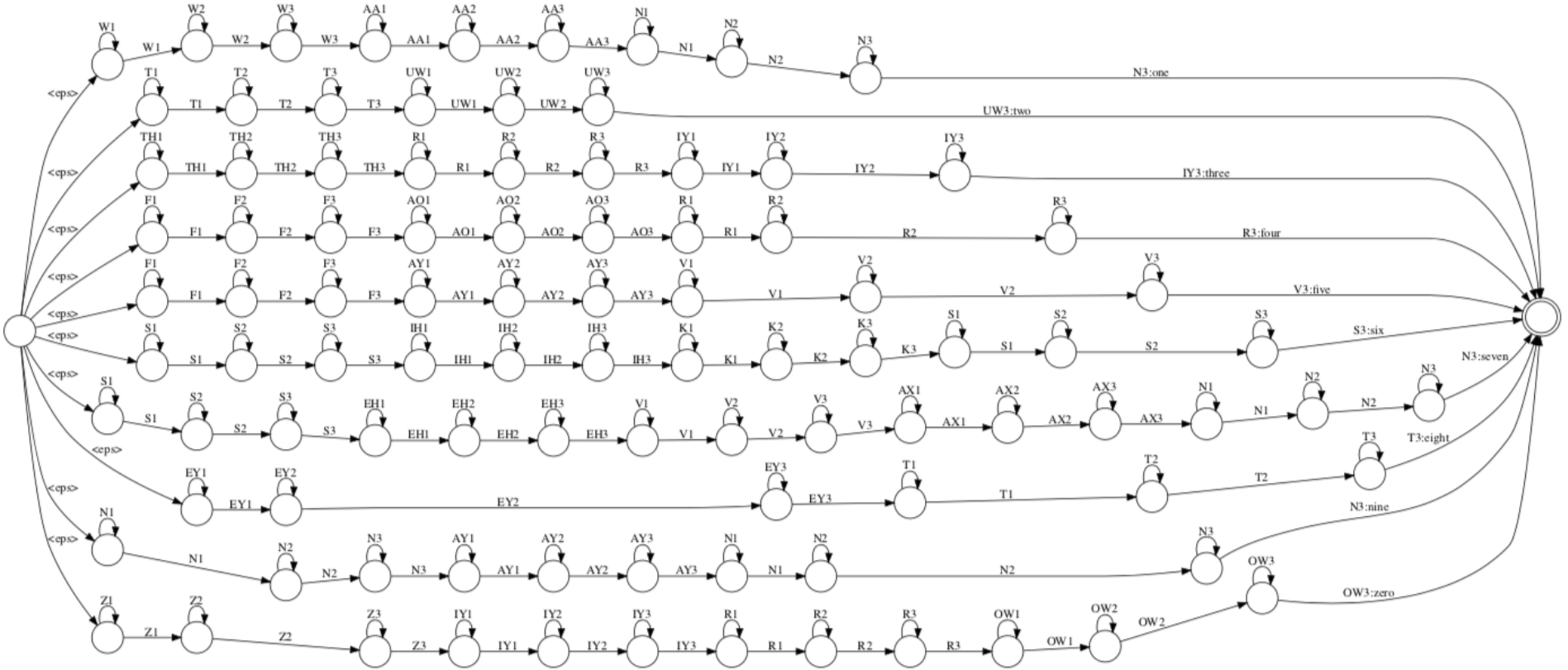


训练

- 现在假设一句话里面包含一个单词，例如one(W AA N)
  - 如何做前向后向训练
  - 如何做Viterbi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？



解码？（思考一下）


<eps>
W1 W2 W3 AA1 AA2 AA3 N1 N2 N3 N3:one
T1 T2 T3 UW1 UW2 UW3 UW3:two
TH1 TH2 TH3 R1 R2 R3 IY1 IY2 IY3 IY3:three
F1 F2 F3 AO1 AO2 AO3 R1 R2 R3 R3:four
F1 F2 F3 AY1 AY2 AY3 V1 V2 V3 V3:five
S1 S2 S3 IH1 IH2 IH3 K1 K2 K3 S1 S2 S3 S3:six
S1 S2 S3 EH1 EH2 EH3 V1 V2 V3 AX1 AX2 AX3 N1 N2 N3 N3:seven
EY1 EY2 EY3 T1 T2 T3 T3:eight
N1 N2 N3 AY1 AY2 AY3 N1 N2 N3 N3:nine
Z1 Z2 Z3 IY1 IY2 IY3 R1 R2 R3 OW1 OW2 OW3 OW3:zero

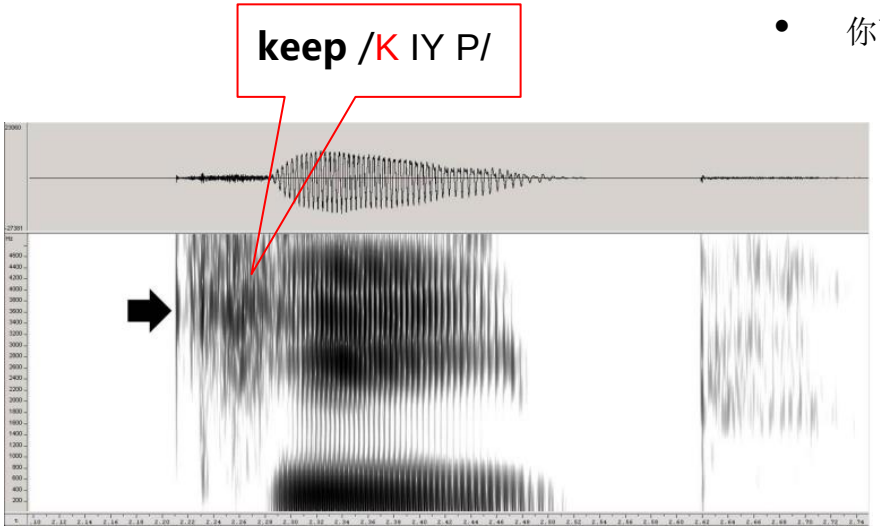
基于三音素的GMM-HMM语音识别系统

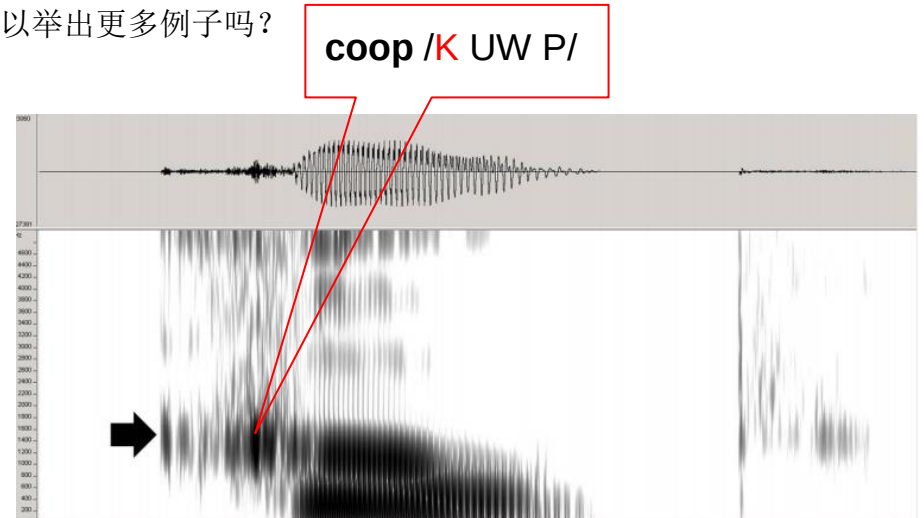
单音素缺点1：建模单元数少

- 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个以上

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
- 你可以举出更多例子吗？





三音素

- 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

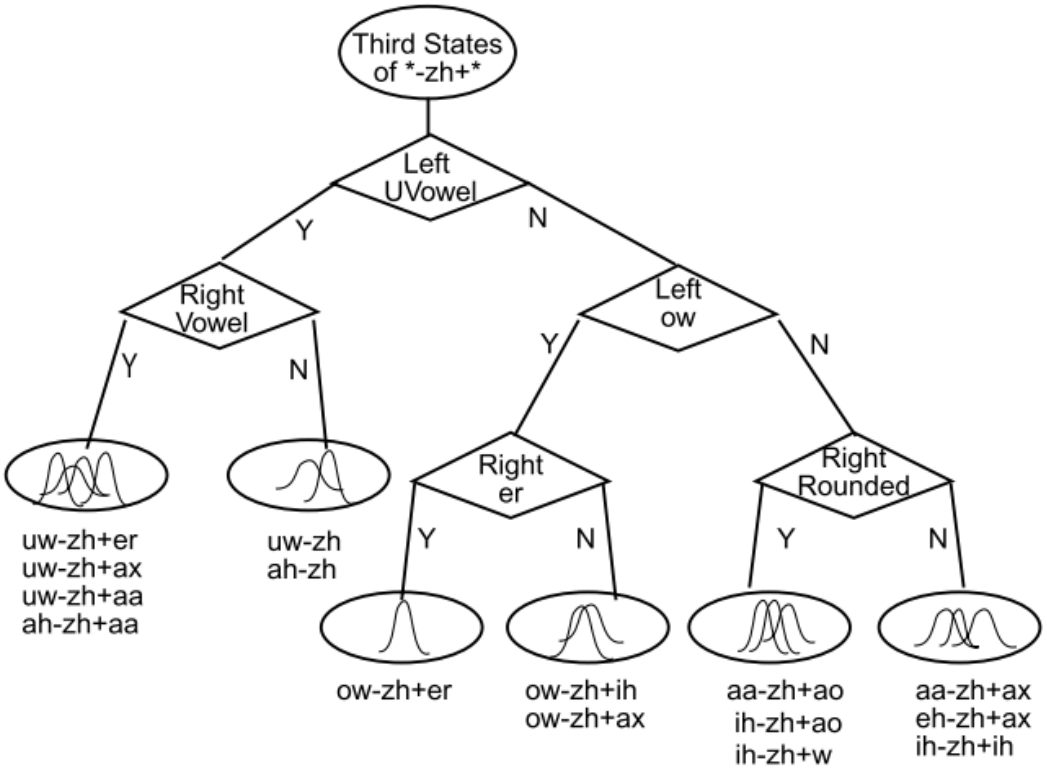
  例如：KEEP　　K IY P　=> #-K+YI, K-IY+P, YI-P+#

- 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



决策树长这样：

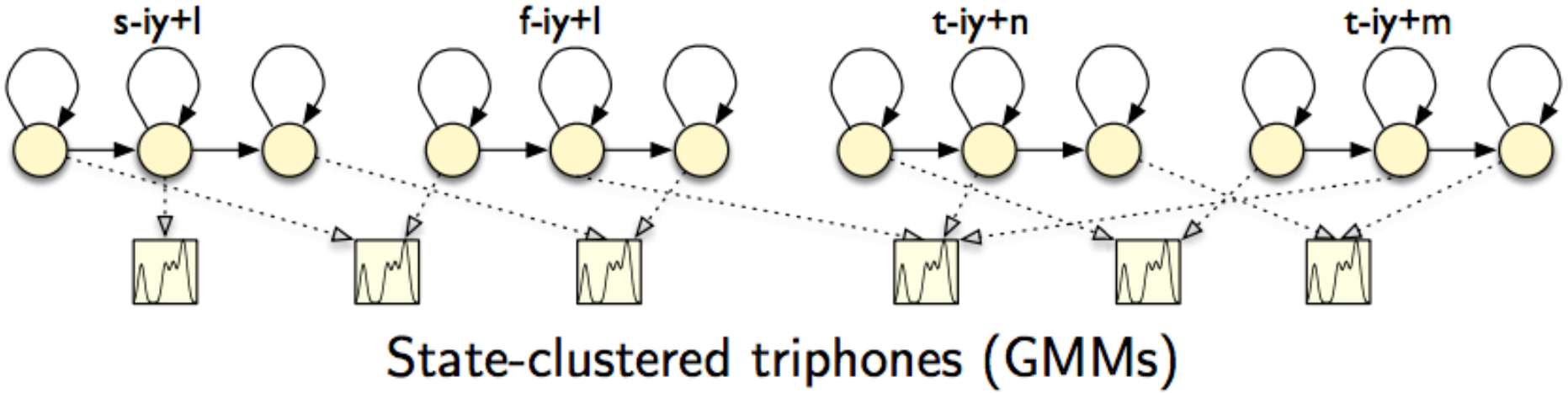
- 二叉树
- 每个非叶子结点上都会有一个问题
- 叶子结点是一个绑定三音素的集合
- 绑定的粒度为状态
  - A-B+C和A-B+D的第1个状态绑定在一起，并不代表其第2/3个状态也要在一起
  - 也就是B的每个状态都有一颗小的决策树
- zh-zh+zh

问题（集）

- 常见问题
  - 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
  - 元音(Vowel): AA AE AH AO AW AX AXR AY EH ER …
- 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



决策树的构建（最优问题）

- 选择哪个问题进行二分类？
- 数据 $S = (x_1, \ldots, x_m) \in (\mathbb{R}^N)^m$
- 模型：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^*

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

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

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

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

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？

语音识别工具 KALDI

- 开源的语音识别工具包 http://kaldi-asr.org/
- 作者: http://www.danielpovey.com/
- 为什么用Kaldi?
  - 语音识别全栈工具
  - 易用，标准数据集标准recipe
  - 流行：社区活跃，几乎所有的语音公司都在用Kaldi
  - 优秀的设计和代码风格
- 单音素系统Toy Demo: https://github.com/kaldi-asr/kaldi/blob/master/egs/yesno/s5/run.sh
- 单音素三音素系统Demo: https://github.com/kaldi-asr/kaldi/blob/master/egs/aishell/s5/run.sh

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
5. Kaldi简介

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
- 为什么我不自己设计作业？
  - 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    //
```

感谢各位聆听！