语音识别基本原理介绍

A Introduction to Basic Principles of Speech Recognition

By wbglearn(吴本谷)，version 1.3

严禁传播1074

2016 年 2 月 2 日

目录

严禁传播 1074

0 写在前面的话

习惯每个文档前总写个这样的前言，一来说明做这件事情的初衷，二来解释下你可以怎么使用这个文档。

那么写这个文档的原因是因为自己从 2013 年开始接触语音识别以来，直到最近的面试才知道自己没有完全知道这些基本的原理。自己之前从互联网上也需要很多的资料，但最终都没有发现这些资料。虽然博客里也零零散散的写了点，总是感觉自己还是欠缺很多东西。写完这个文档后，我还是觉得自己没有完全了解这些，不过我后来会逐步完善这些，争取早点写一个通俗易懂的文档。

第二个就是解释下怎么使用这个文档。当你阅读这个文章前，我希望你有一定的语音识别的基本知识，比如音素，词典，上下文音素、期望最大算法等等。读这篇文章的一些辅助资料我也将给大家。还有就是学习语音识别的一些入门的东西以及深度学习的一些东西也会给大家。此外，后续的一些东西也会给大家。

最后说重点的，这次这个文档是收费的，每个文档都有唯一的编号和版权说明，希望大家理解，大家尽量不要分享给别人，尊重下劳动成果，如果你一定那么做，我也不会知道的。但我相信你不会。如果此文档更新，我会发到大家的邮箱。

Ok，就说这么多，希望你有所收获。

严禁传播 1024

1 语音识别基本原理介绍

1.1 引言

语音识别主要指让机器听懂人说的话，即在各种情况下，准确的识别语音的内容，从而根据其信息，执行人的各种意图。我们可以单一的理解为语音到文字的过程，文字到语音的过程就是语音合成。按识别对象来分，语音识别分为孤立词识别和连续词识别，后续所说的语音识别都指连续词识别。

孤立词识别一般用于命令控制的一些应用中，如语音控制开关等等。

连续词识别应用比较广泛，如讯飞灵犀，百度语音助手，siri，google now，cortana 等等。

说到语音识别，又不得不提到隐马尔科夫模型(HMM)，语音识别主要利用其中的三个重要问题，即评估问题，解码问题和训练问题，对应的算法就是前向算法，维特比算法，前向-后向算法。具体是什么，见 1.2 部分。

然后就是混合高斯模型(GMM)，理解就是几个高斯模型的累加，这样才能表达更为复杂的信号。最后就是深度神经网络，这个是最新的算法，主要是用来替代我们的 GMM。

1.2 隐马尔科夫模型(HMM)

这里具体的隐马尔科夫的数学公式和具体理解，大家可以去看下《HMM 最佳范例》，这里说的非常清楚。HMM 非常广泛，如果你了解了，对你还是有很大好处的。HMM 主要的思想是用迭代的思想来解决动态规划的问题。介绍算法前，先说说 HMM 涉及到的 5 个参数：

2 个状态集合和三个概率集合，2 个状态集合分别为隐藏状态和观察状态，三个概率集合如下：

A 是转移矩阵，$A=[a_{ij}],i,j=1,2,...,N$，其中 $a_{ij}=P\left(q_t=j\,|\,q_{t-1}=i\right),i,j=1,2,...,N$；

π 是初始化时每个状态对应的概率，也称状态占有率，$\pi=\left[\pi_i\right],i=1,2,...N$，$\pi_i=P\left(q_1=i\right),i=1,2,...,N$；

B 是观察向量对应的概率，称为混淆矩阵，这个值是有 GMM 或者 DNN 得到的，$P(o_t\,|\,s^{(i)}),i=1,2,...N$，$b_i\left(o_k\right)=P\left(o_t=v_k\,|\,q_t=i\right),i=1,2,...N$。

下面介绍简单介绍之前说的三大问题。

第一个问题：评估问题。在给定 HMM 模型参数 $\lambda=\left(\pi,A,B\right)$ 以及观测向量 $O=\left\{o_1,o_2,...,o_T\right\}$ 后，我们需要求得模型 λ 产生 O 的似然度 $p\left(O\,|\,\lambda\right)$。这里一般使用前向算法来解决。

第二个问题：解码问题。在给定 $\lambda=(\pi,A,B)$ 以及观测向量 $O=\{o_1,o_2,...,o_T\}$ 后，我们需要搜索出模型 λ 中最可能生成 O 的状态序列 $s^*=\{s_1,s_2,...,s_T\}$。只有这样，我们才可以对 O 进行解码并找出对应于每一帧观测向量的，隐藏的状态转移过程。这里一般使用维特比算法来解决。

第三个问题：训练问题。在给定一串观察序列 O 后，我们还需要通过一定的手段获取模型参数 λ，并使得 $p(O|\lambda)$ 最大化。只有这样，我们才可以在最大似然准则下通过训练的方式得到 HMM 模型的参数。这里一般使用前向-后向算法来解决。

具体使用数学公式来一步一步去做解决这三个问题可以看《声学模型区分性训练及其在 LVCSR 系统的应用.pdf》第二章部分。

下面是一些算法的概述，方便大家利用，其中维特比算法如下：

Algorithm 3.2 Viterbi algorithm for decoding HMM state sequence.

procedure VITERBIDECODE($A=[a_{ij}]$, π, $b_j(\mathbf{o}_t)$)
- ▷ A is the transition probability
- ▷ π is the initial state occupation probability
- ▷ $b_j(\mathbf{o}_t)$ is observation probability given HMM state j for observation vector $\mathbf{o}_t$

$\delta_i(1) \leftarrow \pi_i b_i(\mathbf{o}_1)$ ▷ Initialize at $t=1$

$\psi_i(1) \leftarrow 0$ ▷ Initialize at $t=1$

for $t \leftarrow 2; t \le T; t \leftarrow t+1$ **do** ▷ Forward recursion

  $\delta_j(t) \leftarrow \max_i \delta_i(t-1)a_{ij}b_j(\mathbf{o}_t)$

  $\psi_j(t) \leftarrow \arg\ \max_{1\le i\le N} \delta_i(t-1)a_{ij}$

end for

$P^* \leftarrow \max_{1\le i\le N}[\delta_i(T)]$

$q(T) \leftarrow \max_{1\le i\le N}[\delta_i(T)]$ ▷ Initialize backtracking

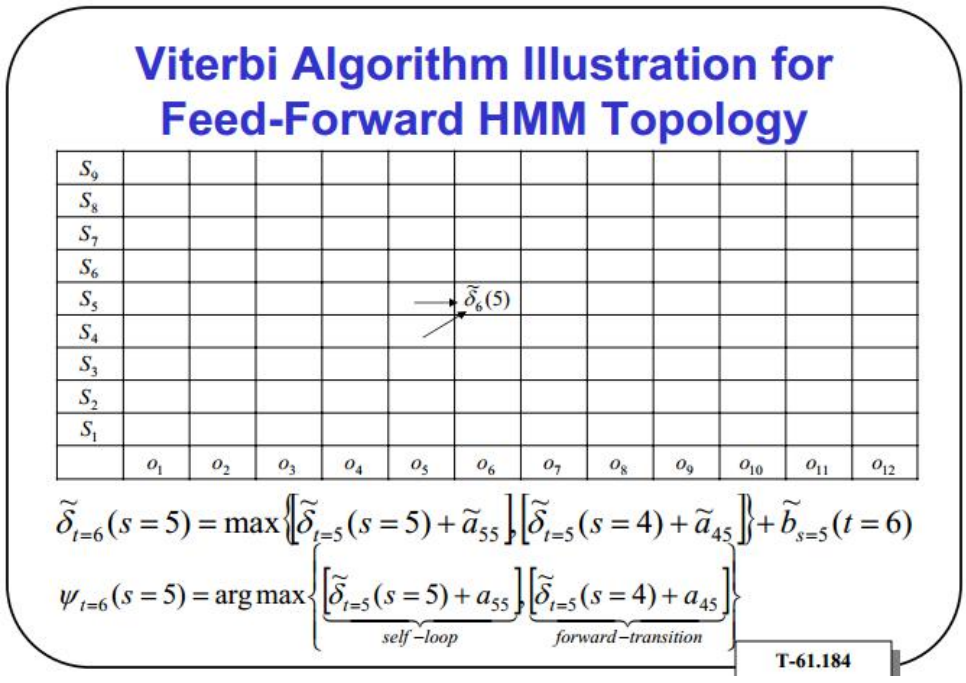
for $t \leftarrow T-1; t \ge 1; t \leftarrow t-1$ **do** ▷ Backward state tracking

  $q^*(t) \leftarrow \psi_{q^*(t+1)}(t+1)$

end for

Return optimal HMM state path $q^*(t)$, $1 \le t \le T$

end procedure



评估算法和 B-W 算法可以去看 http://www.inf.ed.ac.uk/teaching/courses/asr/ 第三章的 ppt。

1.3 物理意义

这里说下语音识别的物理意义。

识别就是给定一段语音，你告诉我它是什么文字。这里我们用数学公式表示出来，给定语音经过我们提取特征后就是一个矩阵 X，我们要得到一个词序列 W，那么最可能的词序列W^*:

$$W^* = \arg\max_W P(W \mid X)$$

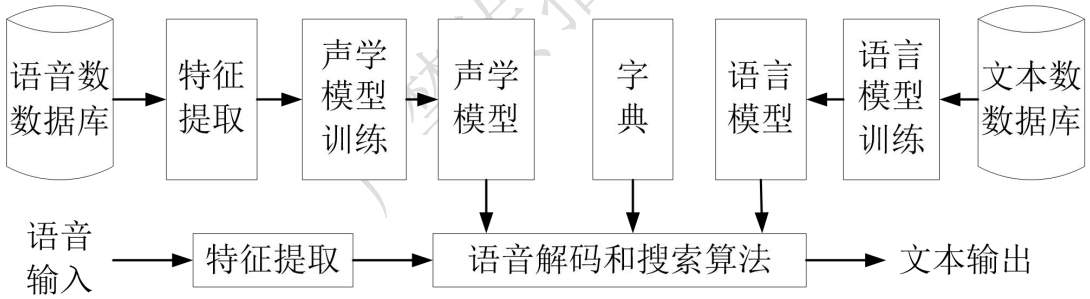
经过贝叶斯公式就变为：

$$W^* = \arg\max_W P(W \mid X) = \arg\max_W \frac{P(X \mid W)P(W)}{P(X)}$$

因为$P(X)$一直是定值，那相当于：

$$W^* = = \arg\max_W P(X \mid W)P(W)$$

这里的$P(X \mid W)$对应我们的声学模型，$P(W)$对应我们的语言模型。所以我们的语音识别框架图也就是：



后面我们将一步一步的把这个图说清楚。

1.4 特征提取

目前主流的特征是 MFCC，当然也有 PLP 或者 pitch 等一些特征，还有深度学习里的 bottleneck 特征和 Mel-fbank 特征。一个 wav 文件的特征一般为 13 维(下图中的 12MFCC+1energy)或者 26 维(下图中的 12MFCC+1energy+12△MFCC+1△energy)或者 39 维(下图中的 12MFCC+1energy+12△MFCC+1△energy+12△△MFCC+1△△energy)乘以帧数的一个矩阵。

MFCC 特征的流程图：

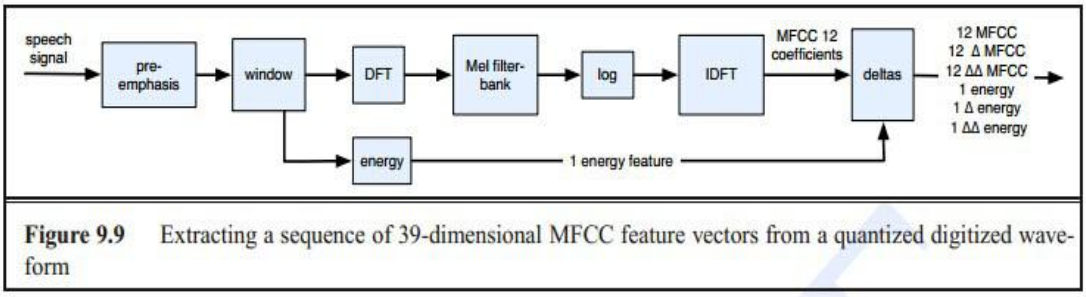


Figure 9.9 Extracting a sequence of 39-dimensional MFCC feature vectors from a quantized digitized waveform

Mel-fbank 就是经过 Mel-filter bank 后的特征，DNN 为什么用这个特征的原因就是 IDFT 这步只是为了去相关，GMM 需要每一维是不相关的，而 DNN 没有这个要求。

MFCC 的每一步对应的公式我也想说清楚啊，这里截图如下：

提取一组 MFCC 特征主要有以下几个步骤：

1.首先对输入的语音信号进行预处理，得到分帧和加窗后的时域信号；

2.对时域信号进行快速傅里叶变换（FFT，Fast Fourier Transform），得到语音信号的频率表达；

3.将得到的线性频率转换为 Mel 频率，转换公式如下

$$Mel(f)=2595\times\log_{10}\left(1+\frac{f}{700}\right) \tag{2.22}$$

或者

$$Mel(f)=1125\times In\left(1+\frac{f}{700}\right) \tag{2.23}$$

4.在 Mel 频率轴上构造 M 个三角带通滤波器组，这 M 个三角滤波器在 Mel 频率尺度上是平均分布的，其定义为

$$H_m(k)=\begin{cases}\dfrac{k-f(m-1)}{f(m)-f(m-1)}, & f(m-1)\le k\le f(m)\\ \dfrac{f(m+1)-k}{f(m+1)-f(m)}, & f(m)<k\le f(m+1)\\ 0, \text{其他}\end{cases} \tag{2.24}$$

其中，m 为滤波器编号$(0\le m\le M-1)$，$f(m-1)$，$f(m)$，$f(m+1)$分别是该滤波器的下限、中心和上线频率。

三角带通滤波器主要有两个目的[12]：一是对频谱进行平滑，消除谐波的影响，突出原始语音的共振峰，因此，以 MFCC 为特征的语音识别系统并不会受到输入语音的音调不同而有所影响；二是降低了信息量。

5.离散余弦变换（DCT，Discrete Cosine Transform）。对每一个滤波器的输出计算其对数能量 E_m，并做 DCT 变换

$$C_d=\sum_{m=0}^{M-1}E_m\cos\left[m\left(k-\frac{1}{2}\right)\frac{\pi}{M}\right], d=0,1\cdots,D-1<M \tag{2.25}$$

其中，D 为 Mel 倒谱系数的阶数，其不同分量对应不同的信息。实验表明，最有用的语音信息包含在 MFCC 分量 C_1 到 C_{12} 之间，最有用的说话人信息包含在 MFCC 分量 C_2 到 C_{16} 之间[13]。

见重邮论文 2.4.2。

1.5 GMM-HMM 声学模型

这部分就是 GMM-HMM。因为上面我们已经得到了特征，而且我们得到的是 39

维特征，如果要有高斯模型的话，一定是多维高斯分布函数。

$$\mathrm{f}\left(x \mid \mu, \Sigma\right)=\frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu)^{T}\Sigma^{-1}(x-\mu)\right)$$

这里的 x 就是输入，μ，Σ 分别表示多维高斯部分函数的均值和方差矩阵。接下来我们把这个函数对应到我们的 HMM 里，

$$b_j(o_t)=\frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(o_t-\mu_j)^{T}\Sigma_j^{-1}(o_t-\mu_j)\right)$$

这个函数里的o_t就是 MFCC 特征。

这里备注一下：因为多维高斯分布里，如果我们输入每一维都是不相互的话，我们的方差矩阵就只有主对角线上有值，协方差矩阵是：

$$\begin{pmatrix}\lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \lambda_n\end{pmatrix}$$

这样计算也挺方便的，只要存储主对角线的元素。
这样的话，我们的多维高斯分布函数可以改变为：

$$b_j(o_t)=\prod_{\mathrm{d}=1}^{D}\frac{1}{\sqrt{2\pi\sigma_{jd}^2}}\exp\left(-\frac{1}{2}[\frac{(o_{td}-\mu_{jd})^2}{\sigma_{jd}^2}]\right)$$

我们真实的计算就是这个函数。这个函数的更新法则就是，首先要计算一个$\xi_t(i)$，这个变量表示在时间 t 为状态 i 的概率，具体计算：

$$\xi_t(i)=P(q_t=i\mid O,\lambda)=\frac{P(q_t=i,O\mid\lambda)}{P(O\mid\lambda)}=\frac{\alpha_t(i)\beta_t(i)}{P(O\mid\lambda)}$$

这个依赖我们的前向和后向概率。
我们的均值更新和方差更新为：

$$\mu_i=\frac{\sum_{t=1}^{T}\xi_t(i)o_t}{\sum_{t=1}^{T}\xi_t(i)}$$

$$\sigma_i^2=\frac{\sum_{t=1}^{T}\xi_t(i)(o_t-\mu_i)(o_t-\mu_i)^{T}}{\sum_{t=1}^{T}\xi_t(i)}$$

这样我们就可以更新我们的均值和方差。

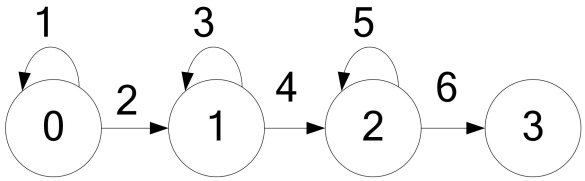
亚杰传播 1074

现在我们来看具体是怎么操作的。

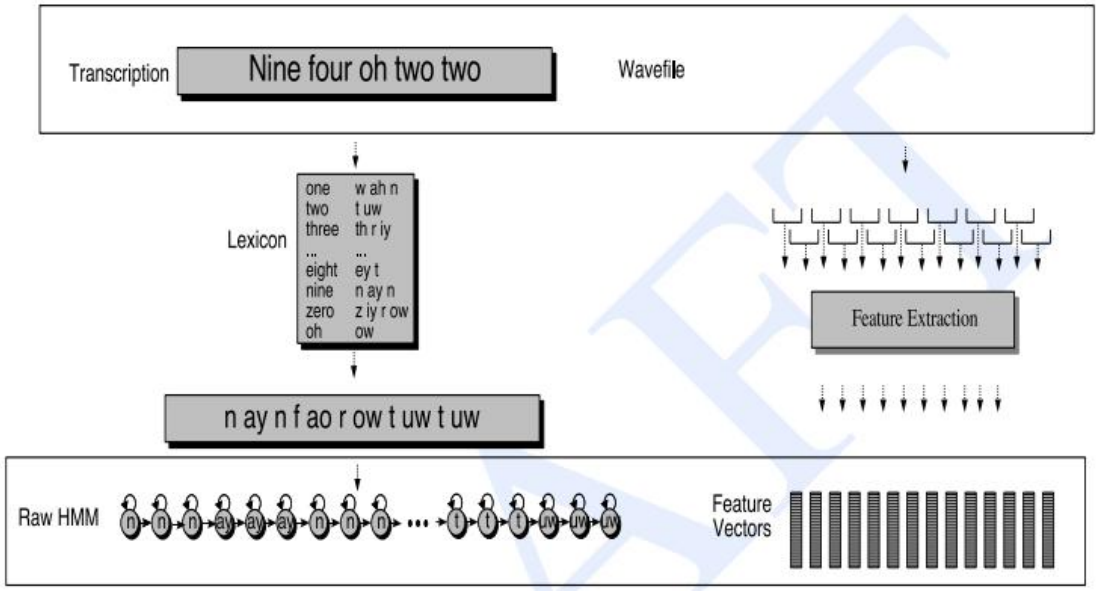
首先，训练的时候我们是以一句话为一个单位的。现在假设我们有一个句子，他标注是词为单位，然后根据词典，映射为：

faem0_si2022 sil w ah dx aw f ix cl d uh sh iy vcl d r ay v f ao sil

一开始训练的时候，我们就要有这个结构。还需要我们定义拓扑结构，比如：



这部分我放一个图大家就明白了。



现在要做的事情就是确定我们特征中的每一帧对应我们的音素的哪一个状态。我们使用我们上面描述的最后那个高斯函数来训练，这里一般使用前向-后向算法就可以得到。很具体的可以参考之前说的 HMM 算法和上面所说的更新准则。下面我讲贴出一个图出来，这个就是具体的过程：

Iterative estimation of HMM parameters using the EM algorithm. At each iteration

E step For all time-state pairs

1. Recursively compute the forward probabilities $\alpha_t(j)$ and backward probabilities $\beta_t(j)$
2. Compute the state occupation probabilities $\gamma_t(j)$ and $\xi_t(i,j)$

M step Based on the estimated state occupation probabilities re-estimate the HMM parameters: mean vectors $\boldsymbol{\mu}_j$, covariance matrices $\boldsymbol{\Sigma}_j$ and transition probabilities a_{ij}

这里的具体的每个变量的计算可以参考《asr03-hmmgmm.pdf》。

当然在 kaldi 里我们是用维特比算法实现的，并且在 kaldi 中是由 fst 来做这些的。迭代到一定次数我们就认为结束。这样我们会得到每一句话的音素和状态对应到我们的特征。但是，很多句子里都含有这个音素怎么办？这时候我们需要把每一个音素对应到的特征放在一起，来训练我们的 HMM 结构，这样我们就得到每个音素的 HMM。我们单音素训练的最终结果也就是要训练每个音素对应的 HMM。至此，单音素训练完毕。

1. **Initialization**: Linearly segment each training utterance into units and HMM states assuming no silence between words (i.e., silence only at the beginning and end of each sentence), a single lexical pronunciation of each word, and a single model for each subword unit. Figure 8.5, iteration 0, illustrates this step for the first few units of one training sentence. Literally we assume every unit is of equal duration initially.
2. **Clustering**: All feature vectors from all segments corresponding to a given state (i) of a given subword unit are partitioned into M clusters using the k-means algorithm. (This step is iterated for all states of all subword units.)
3. **Estimation**: The mean vectors, $\boldsymbol{\mu}_{ik}$, the (diagonal) covariance matrices, $\mathbf{U}_{ik}$, and the mixture weights, c_{ik}, are estimated for each cluster k in state i. (This step is iterated for all states of all subword units.)
4. **Segmentation**: The updated set of subword unit models (based on the estimation of step 3) is used to resegment each training utterance into units and states (via Viterbi decoding). At this point multiple lexical entries can be used for any word in the vocabulary. Figure 8.5 shows the result of this resegmentation step for iterations 1–4 and 10 for one training utterance. It can be seen that by iteration 2 the segmentation into subword units is remarkably stable.
5. **Iteration**: Steps 2–4 are iterated until convergence (i.e., until the overall likelihoods stop increasing).

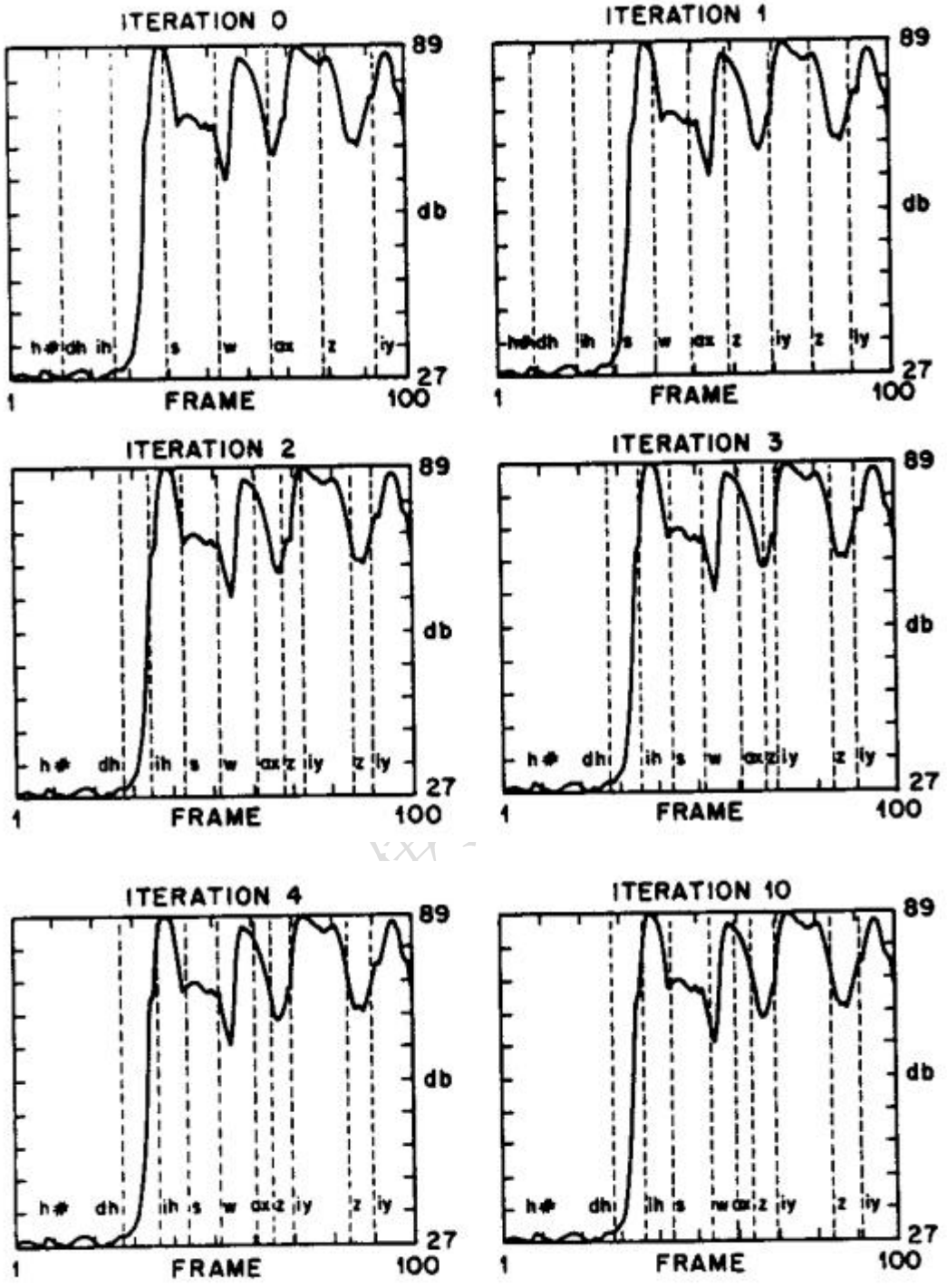


Figure 8.5 Segmentations of a training utterance resulting from the segmental *k*-means training for the first several iterations (after Lee et al. [7]).

总结：

1. 单高斯模型的初始化，这里均值和方差我们使用全局的均值和方差；
2. 生成每个句子的 topo 结构；
3. 首先给每个状态分配相同的帧数，然后开始做对齐；
4. 迭代做对齐，然后计算每个音素对应的 HMM。

2015.8.25 更新：

这次来说清楚，viterbi 对齐的过程，当我们一开始初始化高斯函数的均值和方差时，且已经平均分配到各个状态了。下面将详细的说下是个什么过程：

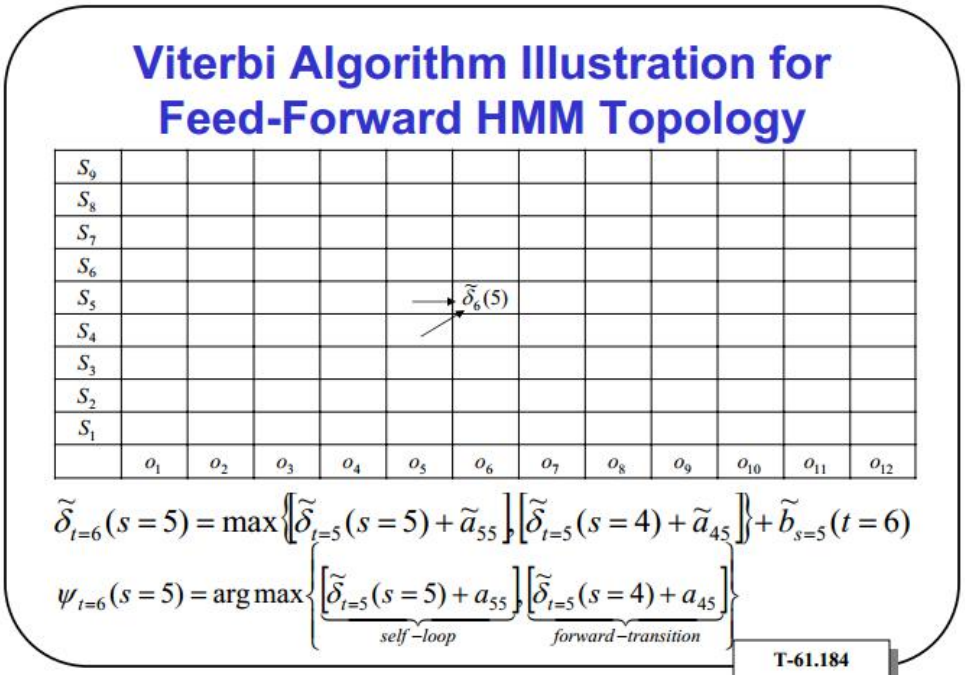
1. 首先，我们把每个状态对应的帧数一起来计算均值和方差，如果某一个状态对应 10 帧，那么就计算这 10 帧一起的均值和方差，这个公式如下：

$$\hat{\mu}_i = \frac{1}{T}\sum_{t=1}^{T} o_t \text{ s.t. } q_t \text{ is state } i$$

$$\hat{\sigma}_j^2 = \frac{1}{T}\sum_{t=1}^{T} (o_t - \mu_i)^2 \text{ s.t. } q_t \text{ is state } i$$

这样，我们得到对应高斯函数的均值和方差，这里可以默认每个状态都是单高斯函数。

2. 然后，再对这一句做对齐的时候，我们这里用 viterbi 对齐，这个的好处就是快的。Kaldi 就是用这种的。具体的如下：



第一帧来时，我们可以默认为第一个状态，那么下一帧来时我们根据 viterbi 更新的公式，可以看下图的算法来得到一条路径，这个路径我们会知道每帧分别对应的是哪个状态。这里你需要更多的思考。

Algorithm 3.2 Viterbi algorithm for decoding HMM state sequence.

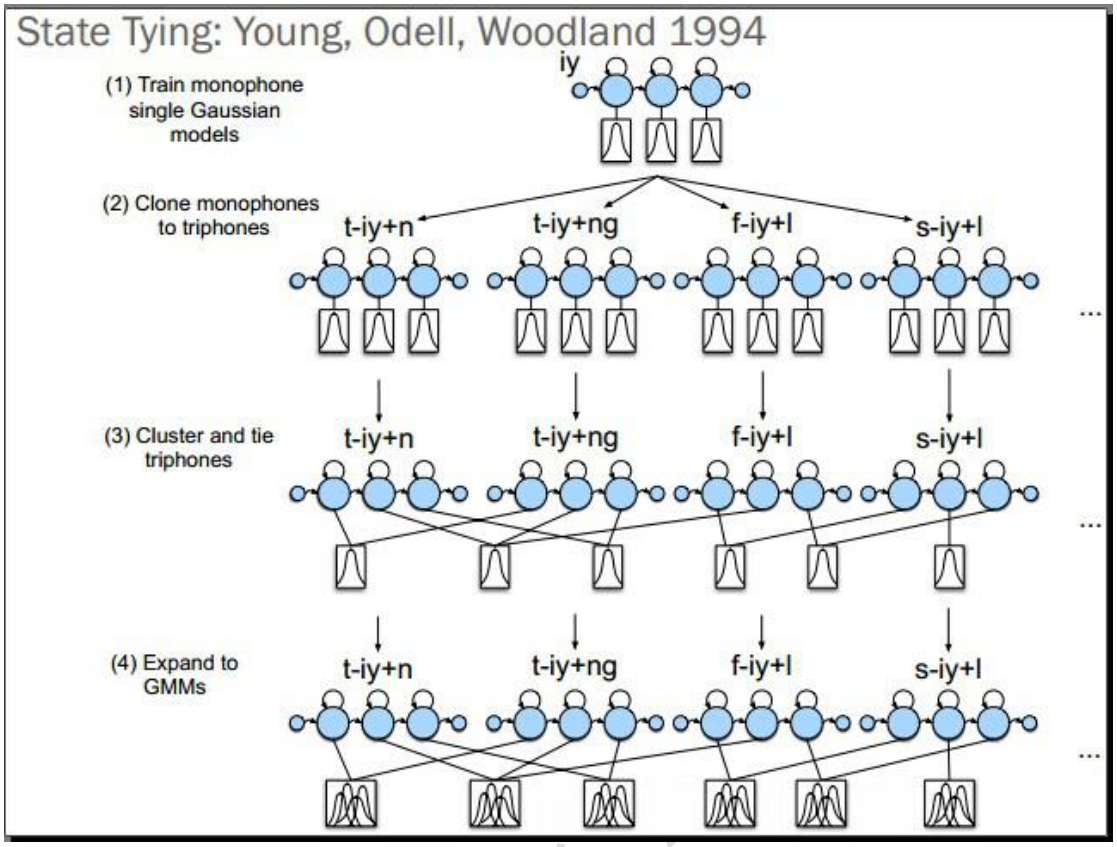
1: **procedure** VITERBIDECODE($A = [a_{ij}], \pi, b_j(\mathbf{o}_t)$)
▷ A is the transition probability
▷ π is the initial state occupation probability
▷ $b_j(\mathbf{o}_t)$ is observation probability given HMM state j for observation vector $\mathbf{o}_t$
2: $\delta_i(1) \leftarrow \pi_i b_i(\mathbf{o}_1)$ ▷ Initialize at $t = 1$
3: $\psi_i(1) \leftarrow 0$ ▷ Initialize at $t = 1$
4: **for** $t \leftarrow 2; t \leq T; t \leftarrow t + 1$ **do** ▷ Forward recursion
5: $\delta_j(t) \leftarrow \max_i \delta_i(t-1)a_{ij}b_j(\mathbf{o}_t)$
6: $\psi_j(t) \leftarrow \arg\ \max_{1 \leq i \leq N} \delta_i(t-1)a_{ij}$
7: **end for**
8: $P^* \leftarrow \max_{1 \leq i \leq N}[\delta_i(T)]$
9: $q(T) \leftarrow \max_{1 \leq i \leq N}[\delta_i(T)]$ ▷ Initialize backtracking
10: **for** $t \leftarrow T-1; t \geq 1; t \leftarrow t-1$ **do** ▷ Backward state tracking
11: $q^*(t) \leftarrow \psi_{q^*(t+1)}(t+1)$
12: **end for**
Return optimal HMM state path $q^*(t)$, $1 \leq t \leq T$
13: **end procedure**

3. 我们知道每帧分别对应的状态后，你可以重复第二步所做的，直到达到一个稍微稳定的对齐结果。

这样，利用单高斯函数来对单音素建模你就清楚了。后面的混合高斯模型无非就是多一个权重的更新，可以去看看对应的公式，然后三音素其实就是状态换了而已。至少这里把强制对齐说的很清楚了。大家看文档多去理解理解，这个说法你不会在任何地方看到的。加油……
注：三音素模型这里没有说的很清楚，如果等我弄清楚后，我会更新版本的，谢谢各位。

三音素训练：
其实整个三音素我们可以通过下图来看：

严禁传播



第一个就是我们之前得到的每个音素的单音素模型。

下面主要分二步：

第一步就是:单音素到三音素的过程。假如我们有 60 个单因素，那么总共有 60 的三次方个三音素，这么多肯定不利于处理，这个怎么实现这种三音素其实我们可以直接利用脚本。这样我们得到一大堆三音素，但这些转移概率矩阵是一样的。

第二步就是聚类，这里主要使用决策树来完成。在 htk 或者 kaldi 里也不需要我们去做，这个我们只要设定一些参数就可以。这个目的就是减少计算。具体决策树的部分我就不展开了，可以去看《Speech and Language Processing.pdf》第 10.3 部分。

这里我们复制每个单音素模型，对每一个三音素做复制，但转移矩阵 A 是不复制的，但是一个单音素的所有三音素复制都做绑定，然后我们使用 EM 算法迭代重新训练三音素模型。

现在对于每一个单音素，我们用聚类算法来对所有的上下文相关的三音素做聚类，得到绑定状态聚类的一个集合。一个典型的状态被选择作为整个类的代表，剩下的都跟它绑定，这个类就是 senone。

下面介绍混合高斯模型以及它的更新法则。混合高斯模型是：

$$\mathrm{f}\left(x \mid \mu, \Sigma\right)=\sum_{k=1}^{M} c_k \frac{1}{\sqrt{2\pi\left|\Sigma_k\right|}} \exp\left(\left(x-\mu_k\right)^T \Sigma^{-1}\left(x-\mu_k\right)\right)$$

那么对应到我们的 HMM 里就是：

$$b_j(o_t)=\sum_{m=1}^{M}c_{jm}\frac{1}{\sqrt{2\pi|\Sigma_{jm}|}}\exp\left((o_t-\mu_{jm})^T\Sigma_{jm}^{-1}(o_t-\mu_{jm})\right)$$

更新的时候我们要定义 $\xi_{tm}(j)$ 为在时间 t 是状态 j 的第 m 个高斯分量的概率：

$$\xi_{tm}(j)=\frac{\sum_{i=1}N\alpha_{t-1}(j)a_{ij}c_{jm}b_{jm}(o_t)\beta_t(j)}{\alpha_T(F)}$$

均值、方差和权重的更新：

$$\mu_{im}=\frac{\sum_{t=1}^{T}\xi_{tm}(i)o_t}{\sum_{t=1}^{T}\sum_{\mathrm{m}=1}^{M}\xi_{tm}(i)}$$

$$c_{im}=\frac{\sum_{t=1}^{T}\xi_{tm}(i)}{\sum_{t=1}^{T}\sum_{k=1}^{M}\xi_{tk}(i)}$$

$$\Sigma_{im}=\frac{\sum_{t=1}^{T}\xi_t(i)(o_t-\mu_{im})(o_t-\mu_{im})^T}{\sum_{t=1}^{T}\sum_{k=1}^{M}\xi_{tk}(i)}$$

以上就是混合高斯模型的部分。

下面为更加具体的一个流程：

1.我们使用标准的 embedded training 来训练上下文独立模型，使用 EM 来迭代，得到每个单音素的每个子音素的单独的单个高斯模型；

2.我们来 clone 每一个单音素模型，比如用 3 个子状态的高斯来做复制，每个对应一个三音素转移 素的 tied together；我们来做 EM，和重新训练三音素高斯。至此，对每个单音素，我们使用聚类算法来聚所有上下文相关三音素来得到 tied 状态类的一个集合。

我们使用相同的复制方法来学习混合高斯。

1.对于每一个 tied 三音素状态，我们使用 embedded training 来使用多次迭代 EM 来计算单个混合高斯模型；

2.我们对每个状态分裂 2 个相同的高斯，通过静音来改变这些值，和继续运行 EM 来重新训练这些值；

3.然后我们把每一个分裂成两个高斯，这样就是 4 个了，改变下，重新训练。直到在每个状态每个观察值有一定数量的混合高斯。

至此，三音素训练完毕。我们会得到很多个三音素的 HMM。

广东先播 1074

下面这个是英文的一个描述，先看，我在后面将解释这个。

(1) A flat-start monophone set is created in which each base phone is a monophone single-Gaussian HMM with means and covariances equal to the mean and covariance of the training data.

(2) The parameters of the Gaussian monophones are re-estimated using 3 or 4 iterations of EM.

(3) Each single Gaussian monophone q is cloned once for each distinct triphone $x - q + y$ that appears in the training data.

(4) The resulting set of *training-data* triphones is again re-estimated using EM and the state occupation counts of the last iteration are saved.

(5) A decision tree is created for each state in each base phone, the training-data triphones are mapped into a smaller set of tied-state triphones and iteratively re-estimated using EM.

The final result is the required tied-state context-dependent acoustic model set.

第一步和第二步就不在解释了，第三步由单音素到三音素的复制，首先在第一步单音素阶段里已经得到每个句子的单音素序列，再根据下图里的 Cross Word 的方法把单音素映射为三音素。这个也就是一个脚本的事情。HTK 里 HHED 就是干这个事情的。这样去训练就对应的是三音素的对齐了。剩下就是决策树聚类了，聚类一般分为二种方式，一种是 HTK 里的那种带问题的，比如前鼻音、辅音等等，还有一种就是根据分裂的代价函数来决定。一般使用代价函数来自动分类，代价函数将贴在下图中。

Within word context dependency (triphone):

speech task = / **sil s+p s-p+iy p-iy+ch iy-ch t+ae t-ae+s ae-s+k s-k sil**/

Cross word context dependency (triphone):

speech task = / **sil sil-s+p s-p+iy p-iy+ch iy-ch+t ch-t+ae t-ae+s ae-s+k s-k+sil sil** /

图 3 两种方式的三音素

Assuming tying state does not change frame/state alignment

$$\mathcal{L}(\mathbf{S}) = \sum_{f \in F} \sum_{s \in \mathbf{S}} \log p(\mathbf{o}; \boldsymbol{\mu}(\mathbf{S}), \boldsymbol{\Sigma}(\mathbf{S})) \gamma_s(\mathbf{o}_f)$$

If the state output distribution is Gaussian

$$\mathcal{L}(\mathbf{S}) = -\frac{1}{2}(\log[(2\pi)^n |\boldsymbol{\Sigma}(\mathbf{S})|] + n) \sum_{f \in F} \sum_{s \in S} \gamma_s(\mathbf{o}_f)$$

Find the question maximising

$$\Delta \mathcal{L}_q = \mathcal{L}(\mathbf{S}_y(q)) + \mathcal{L}(\mathbf{S}_n(q)) - \mathcal{L}(\mathbf{S})$$

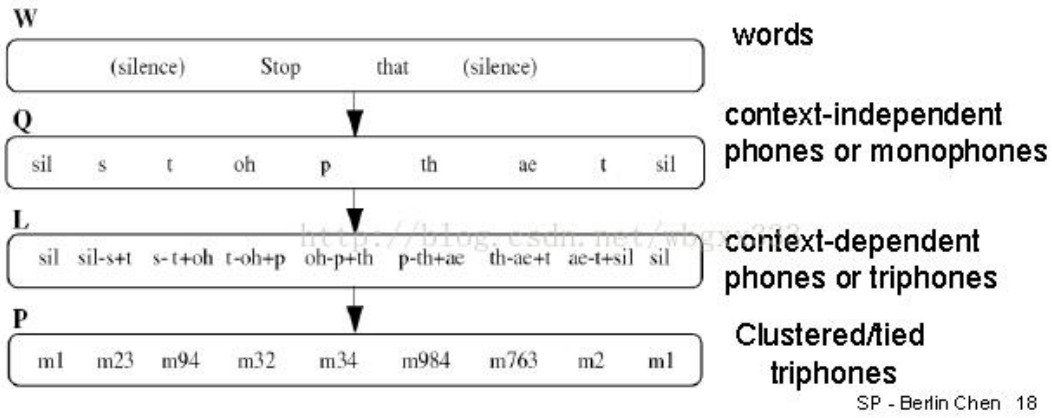
图 4 决策树的代价函数

1. Initialization: pull all data together and calculate likelihood of the root node
2. For each leaf node, do
   - 2.1 For each question, calculate likelihood increase after split
   - 2.2 Select the question with maximum likelihood increase
   - 2.3 If the likelihood increase is higher than the predefined threshold, perform split using the selected question
3. Finally, check the occupancy counts of each leaf node and merge the nodes with small occupancy counts

图 5 决策树的流程

补充：

三音素的训练其实跟单音素的训练一样，只是把单音素换成了三音素。做复制是为了考虑音素上下文的信息，和聚类仅仅为了降低三音素的数量。最后的结果也是每句话对应三音素的标注，学习到每个三音素的 HMM。比如：



像 stop that 这种标注一开始在单音素里是第二行对应的，到三音素里就是第三行里的标注，我们对其训练和对齐，最后一行是做聚类和绑定的，也是最终我们需要的三音素集合。最终的就是最后一行的三音素的 HMM。

总结下具体步骤：

1.建立一个单音素系统(Build a monophone system)

- Single Gaussian component flat start (use first pronunciation)
- Realign training transcriptions (choose correct pronunciation)

2.Build a cross-word triphone system

- Clone monophone to triphone
- Train 1-2 iterations using unclustered triphone models
- Perform decision tree state clustering
- Perform Baum-Welch re-estimation
- Iterative mixture splitting

3.Rebuild decision tree

- 2-model re-estimate single Gaussian unclustered triphone system using final system from 2 as alignment model
- Perform decision tree state clustering
- Perform Baum-Welch re-estimation
- Iterative mixture splitting

1.6 语言模型

备注：这里我们就只介绍统计语言模型中最常用的 n-gram。主要是因为这个用的最广泛。

给定一个词序列 W，那么这个 $P(W)$ 的计算可以表示为很多个条件概率的乘积。

$$
\begin{aligned}
P(W) &= P(w_1, w_2, ..., w_m) \\
&= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2)..P(w_n \mid w_1, w_2, ..., w_{m+1}) \\
&= P(w_1)\prod_{i=2}^{m} P(w_i \mid w_1, w_2, ..., w_{i-1})
\end{aligned}
$$

这个就是我们常说的 n-gram。当然我们平时用的最多的就是 3-gram，最多就是 4-gram。如果是 3-gram，那么我们就需要计算 $P(w_i \mid w_{i-1}, w_{i-2})$：

$$
P(w_i \mid w_{i-1}, w_{i-2}) = \frac{P(w_i w_{i-1} w_{i-2})}{P(w_{i-1} w_{i-2})}
$$

这个我们就很好的去求了，我们需要统计二个词和三个词的概率，然后就解决了。当然，肯定也没那么简单，这里需要考虑平滑处理或者一些零概率的组合等等。

1.7 DNN-HMM 部分

这里假设我们已经得到 mfcc 特征或者 fbank 特征。

因为 DNN-HMM 的基础是从 GMM-HMM 开始的，假设我们也有一个训练好的三音素的 GMM-HMM 模型。

DBN 预训练阶段：

我们首先来看下这个图：

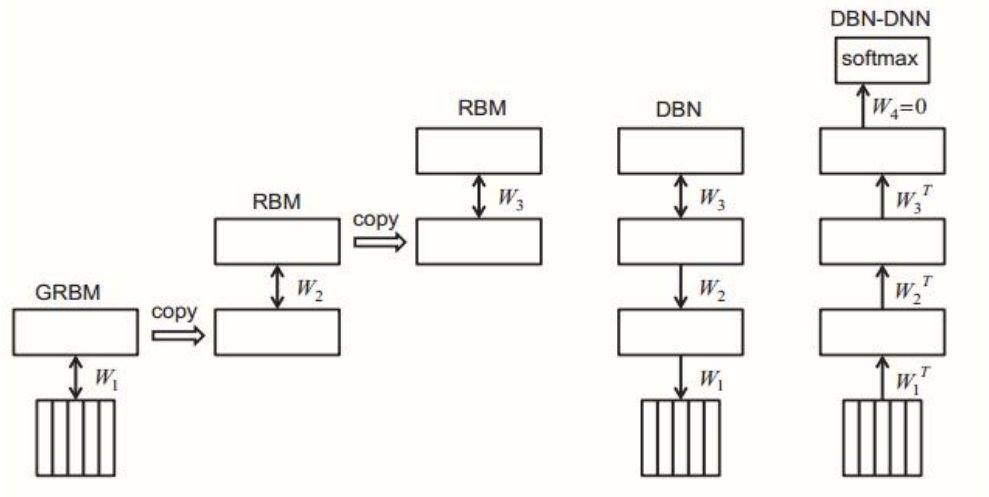


Fig. 1. The sequence of operations used to create a DBN with three hidden layers and to convert it to a pre-trained DBN-DNN. First a GRBM is trained to model a window of frames of real-valued acoustic coefficients. Then the states of the binary hidden units of the GRBM are used as data for training an RBM. This is repeated to create as many hidden layers as desired. Then the stack of RBMs is converted to a single generative model, a DBN, by replacing the undirected connections of the lower level RBMs by top-down, directed connections. Finally, a pre-trained DBN-DNN is created by adding a "softmax" output layer that contains one unit for each possible state of each HMM. The DBN-DNN is then discriminatively trained to predict the HMM state corresponding to the central frame of the input window in a forced alignment.

DNN 的输入是 11*40（40 维 fbank），也就是说我们第一层的节点数应该为 11*40 个，假设输出层的节点为 2048。由于我们特征是实值，我们使用的是 GRBM(高斯到二值)，根据初始值（包括偏移量和权值）来计算输出层的输出，然后根据 RBM 训练算法，得到这一层的权值和标签。以后每层都是 RBM(二值到二值)，每层根据 RBM 训练算法，得到输出。注意：由于这部分是无监督训练，我们的输出不需要与标签对比，根据对比散度算法，有输入层到隐层，然后再到输入层，这样就会有个目标函数，也就是自己跟自己对比。

Fine-tuning 阶段：

当预训练完成后，我们添加一层随机初始化的 softmax 层。（softmax 的输出层节点对应 HMM 的状态，这里对应我们 GMM-HMM 训练好的状态数）。然后我们使用后向传播算法来更新我们的每一层的权值。

这里，也可以这么看，把这些 RBM 堆积起来看成一个几层的 MLP 网络。这样 MLP 的初始权值就是我们之前 RBM 得到的，这里就是加了一层 softmax 来对应 HMM 的状态数。这里还有转换的一步，就是 DNN 得到的是后验概率 $p(s|x)$，使用贝叶斯公式， $p(x|s)=p(s|x)*p(x)/p(s)$

这里的 $p(x)$是常值，$p(s)$是状态的先验，称为尺度似然比。$p(x|s)$对应观察概率，对应 HMM 的输入。

P 的求法是 n(s)/n，n(s)就是所有语音训练数据中根据 alignment 统计的状态 s 的个数,分母 n 就是所有语音训练数据的帧数。

然后就是迭代去训练，得到神经网络的权值和偏置量。

下面这个中文的图可能更加便于大家的理解。

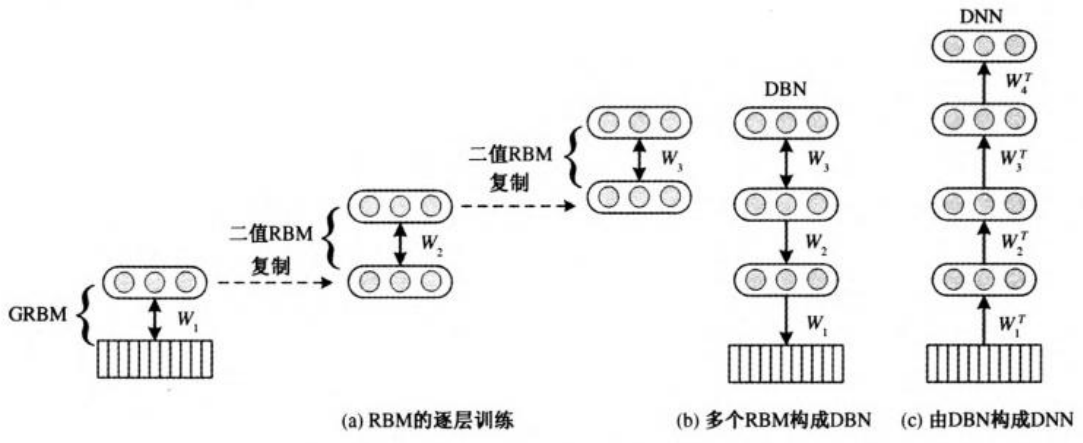


图 2　利用 RBM 构造用于音素识别的深层神经网络

稍微更加具体的步骤，你可以结合我说的和《一种基于 RBM 的深层神经网络音素识别方法.pdf》来看。

这里还是贴一下：

步骤 1　RBM 的逐层训练

①为了充分利用语音的上下文信息，将当前帧前后的多帧 MFCC 串接后作为 GRBM 的输入，使用 CD_1 算法训练 GRBM，得到第 1 层 GRBM 的权值 W_1；使用 GRBM 的输入和 W_1，得到 GRBM 的隐藏单元的状态 H_1。

②使用 H_1 作为下一个 RBM 的输入数据，使用 CD_1 算法进行训练，得到该层 RBM 的权值 W_2；使用 H_1 和 W_2，得到该层 RBM 的隐藏单元状态 H_2。

③重复执行②，直至达到所需要的层数。

步骤 2　利用多个 RBM 搭建 DBN

按照步骤 1 中 RBM 训练的先后次序，自底向上将多个 RBM 依次通过权值连接起来，构成 DBN。在这个 DBN 中，最上面两层（最后训练得到的 RBM）之间的连接是无方向的，DBN 的其它层是自顶向下的有向连接。（使用该模型生成数据时，顶层 RBM 执行多步 Gibbs 交替采样，然后自顶向下通过其余各层，得到生成的数据。）

步骤 3　利用 DBN 构造用于语音识别的深层神经网络

获得 DBN 之后，在其顶层之上，再增加一个“软最大化（softmax）”输出层，输出层的每个节点对应 HMM 中的一个状态。此时的网络称为经过预训练的深层神经网络（deep neural network，DNN）。

步骤 4　DNN 权值的精调整

通过 RBM 逐层预训练得到的 DNN 还需要进行区分性训练（精调整），才能够针对输入的声学特征生成音素状态的后验概率 p（HMM 状态|声学特征）。与预训练的无监督学习不同，精调整是有监督的学习。针对多帧的输入声学特征，取其中间帧所对应的 HMM 状态作为 DNN 的学习目标，在预训练获得的网络权值的基础上，采用某种训练准则，通过反向传播算法获得神经网络的最终权值。

经过精调整之后，DNN 输出形式为 p（HMM 状态|声学特征）的概率。为了在 HMM 框架下进行 Viterbi 解码，还需要将 DNN 的输出转换为似然比 p（声学特征|HMM 状态）。根据贝叶斯公式可知：

$$p(\text{声学特征}\mid HMM\ \text{状态}) = \frac{p(HMM\ \text{状态}\mid \text{声学特征})}{p(HMM\ \text{状态})} p(\text{声学特征}) \tag{14}$$

其中，p（HMM 状态）可以通过训练集中的 HMM 状态进行强制对齐后统计获得或者简单的认为其是一个固定值，p（声学特征）是未知的，但是它对于所有状态都是相同的，因此不会影响后续 Viterbi 解码的效果。将经过（14）式计算得到的状态似然比送入 Viterbi 解码器[1]进行解码，即可以得到音素序列。

具体算法：

Algorithm 1 Main Steps to Train CD-DNN-HMMs

1: Train a best tied-state CD-GMM-HMM system where state tying is determined based on the data-driven decision tree. Denote the CD-GMM-HMM gmm-hmm.

2: Parse gmm-hmm and give each senone name an ordered senoneid starting from 0. The senoneid will be served as the training label for DNN fine-tuning.

3: Parse gmm-hmm and generate a mapping from each physical tri-phone state (e.g., b-ah+t.s2) to the corresponding senoneid. Denote this mapping state2id.

4: Convert gmm-hmm to the corresponding CD-DNN-HMM dnn-hmm1 by borrowing the tri-phone and senone structure as well as the transition probabilities from gmm-hmm.

5: Pre-train each layer in the DNN bottom-up layer by layer and call the result ptdnn.

6: Use gmm-hmm to generate a state-level alignment on the training set. Denote the alignment align-raw.

7: Convert align-raw to align where each physical triphone state is converted to senoneid.

8: Use the senoneid associated with each frame in align to fine-tune the DBN using back-propagation or other approaches, starting from ptdnn. Denote the DBN dnn.

9: Estimate the prior probability p(si) = n(si)/n, where n(si) is the number of frames associated with senone si in align and n is the total number of frames.

10: Re-estimate the transition probabilities using dnn and dnn-hmm1 to maximize the likelihood of observing the features. Denote the new CD-DNN-HMM dnn-hmm2.

11: Exit if no recognition accuracy improvement is observed in the development set; Otherwise use dnn and dnn-hmm2 to generate a new state-level alignment align-raw on the training set and go to Step 7.

1.8 区分性训练

区分性训练大家去看些资料就知道他对于提高我们的识别率还是很显著的。区分性训练是为了弥补最大似然估计的假设，更加符合现实情况。具体的大家可以看下论文：声学模型区分性训练及其在 LVCSR 系统的应用.pdf 等等。

严禁转载 1074

参考文献和你需要读的资料有：

具体推荐三个论文和一本书：

1. Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition
2. Deep Neural Networks for Acoustic Modeling in Speech Recognition
3. 一种基于 RBM 的深层神经网络音素识别方法.pdf
4. Speech and Language Processing.pdf
5. 声学模型区分性训练及其在 LVCSR 系统的应用.pdf
6. The Application of Hidden Markov Models in speech recognition.pdf

2 附录

这部分主要是收录网上有些人的语音识别理解。

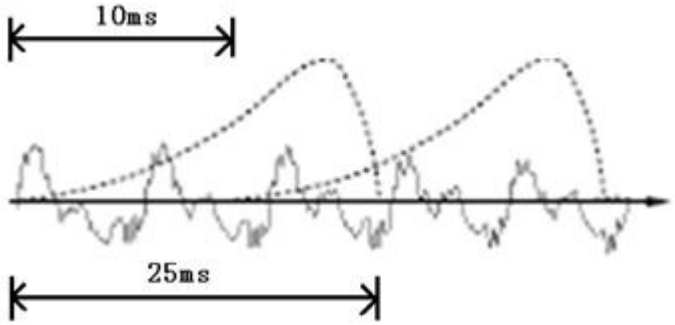
2.1 来自知乎

网址：http://www.zhihu.com/question/20398418

简要给大家介绍一下语音怎么变文字的吧。

首先说一下作为输入的时域波形。我们知道声音实际上是一种波。常见的 mp3、wmv 等格式都是压缩格式，必须转成非压缩的纯波形文件，比如 Windows PCM 文件，即 wav 文件来处理。wav 文件里存储的除了一个文件头以外，就是声音波形的一个个点了。采样率越大，每毫秒语音中包含的点的个数就越多。另外声音有单通道双通道之分，还有四通道的等等。对语音识别任务来说，单通道就足够了，多了浪费，因此一般要把声音转成单通道的来处理。下图是一个波形的示例。

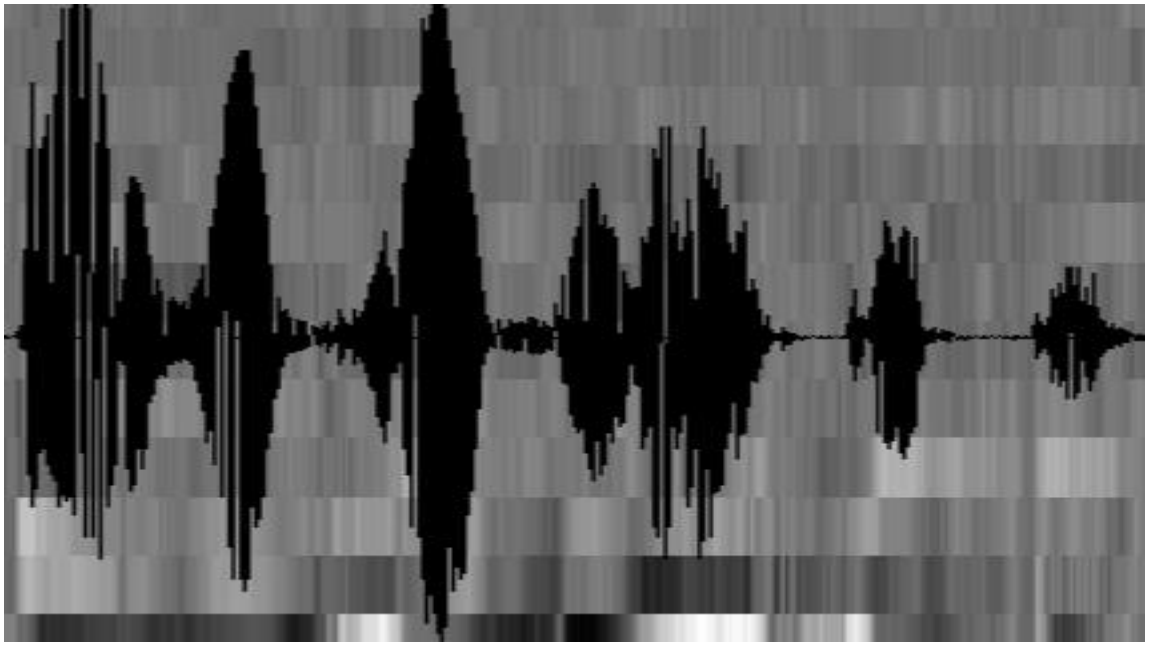
另外，通常还需要做个 VAD 处理，也就是把首尾端的静音切除，降低对后续步骤造成的干扰，这需要用到信号处理的一些技术。
时域的波形必须要分帧，也就是把波形切开成一小段一小段，每小段称为一帧。分帧操作通常使用移动窗函数来实现，分帧之前还要做一些预加重等操作，这里不详述。帧与帧之间是有交叠的，就像下图这样：



图中，每帧的长度为25毫秒，每两帧之间有25-10=15毫秒的交叠。我们称为以帧长25ms、帧移10ms 分帧。

分帧后，语音就变成了很多小段。但波形在时域上几乎没有描述能力，因此必须将波形作变换。常见的一种变换方法是提取 MFCC 特征，把每一帧波形变成一个12维向量。这12个点是根据人耳的生理特性提取的，可以理解为这12个点包含了这帧语音的内容信息。这个过程叫做声学特征提取。实际应用中，这一步有很多细节，比如差分、均值方差规整、高斯化、降维去冗余等，声学特征也不止有 MFCC 这一种，具体就不详述了。

至此，声音就成了一个12行（假设声学特征是12维）、N 列的一个矩阵，称之为观察序列，这里 N 为总帧数。观察序列如下图所示，图中，每一帧都用一个12维的向量表示，色块的颜色深浅表示向量值的大小。

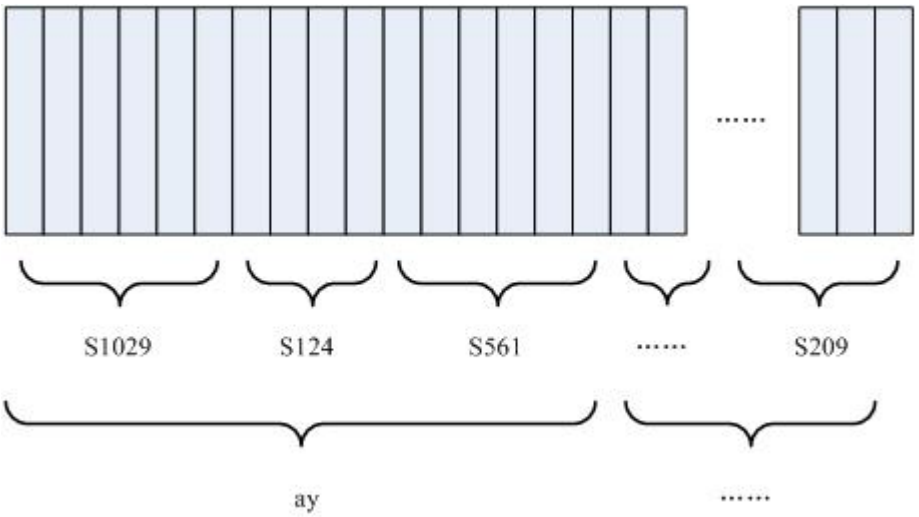


接下来就要介绍怎样把这个矩阵变成文本了。首先要介绍三个概念：

1　单词：英语中就是单词，汉语中是汉字。
2　音素：单词的发音由音素构成。对英语，一种常用的音素集是卡内基梅隆大学的一套由39个音素构成的音素集，参见 The CMU Pronouncing Dictionary[7]。汉语一般直接用全部声母和韵母作为音素集，另外汉语识别还分有调无调，不详述。
3　状态：比音素更细致的语音单位。通常一个音素由3个状态构成。

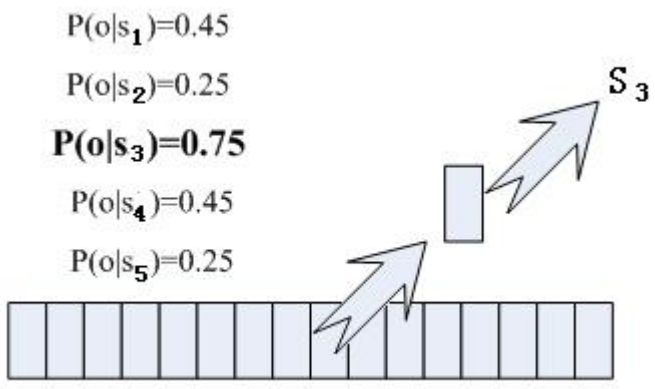
严禁作商业用途

语音识别是怎么工作的呢？实际上一点都不神秘，无非是：
第一步，把帧识别成状态（难点）。
第二步，把状态组合成音素。
第三步，把音素组合成单词。
如下图所示：



图中，每个小竖条代表一帧，若干帧语音对应一个状态，每三个状态组合成一个音素，若干个音素组合成一个单词。也就是说，只要知道每帧语音对应哪个状态了，语音识别的结果也就出来了。

那每帧音素对应哪个状态呢？有个容易想到的办法，看某帧对应哪个状态的概率最大，那这帧就属于哪个状态，这叫做“最大似然”。比如下面的示意图，这帧对应 S3状态的概率最大，因此就让这帧属于 S3状态。



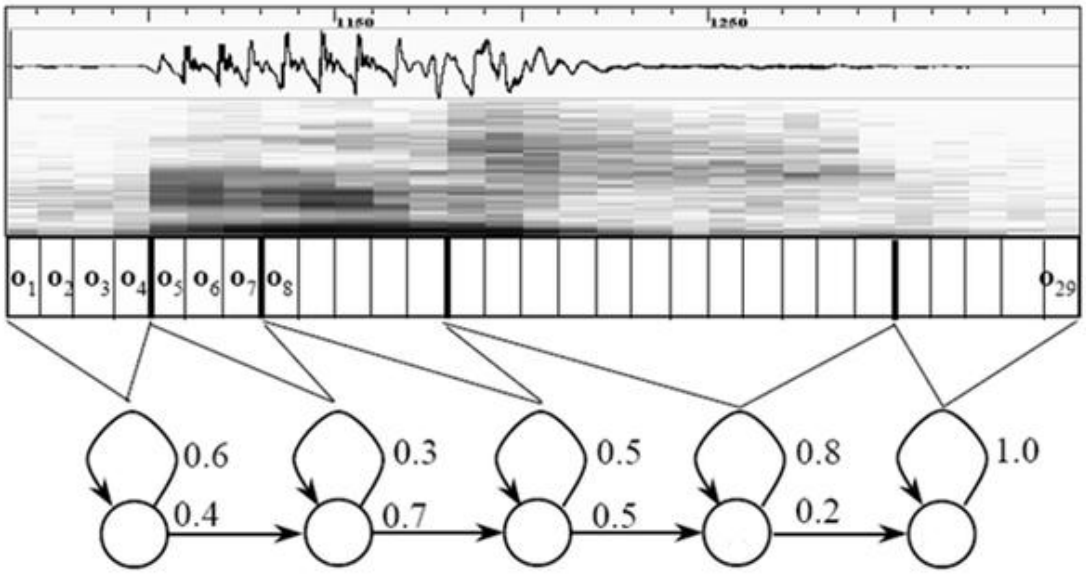
那这些用到的概率从哪里读取呢？有个叫“声学模型”的东西，里面存了一大堆参数，通过这些参数，就可以知道帧和状态对应的概率。声学模型是使用巨大数量的语音数据训练出来的，训练的方法比较繁琐，这里不讲。

但这样做有一个问题：每一帧都会得到一个状态号，最后整个语音就会得到一堆乱七八糟的状态号，相邻两帧间的状态号基本都不相同。假设语音有1000帧，每帧对应1个状态，每3个状态组合成一个音素，那么大概会组合成300个音素，但这段语音其实根本没有这么多音素。实际上如果真这么做，得到的状态号可能根本无法组合成音素。实际上，相邻帧的状态应该大多数都是相同的才合理，因为每帧很短。

解决这个问题的常用方法就是使用隐马尔可夫模型（Hidden Markov Model，HMM）。这东西听起来很复杂，实际上没什么神秘的，无非是：
第一步，构建一个状态网络。
第二步，从状态网络中寻找与声音最匹配的路径。

这样就把结果限制在预先设定的网络中，避免了刚才说到的问题，当然也带来一个局限，比如你设定的网络里只包含了“今天晴天”和“今天下雨”两个句子的状态路径，那么不管说些什么，识别出的结果必然是这两个句子中的一句。

具体是这样的，首先构造单词级网络，然后展开成音素网络，然后展开成状态网络。然后在状态网络中搜索一条最佳路径，这条路径和语音之间的概率（称之为累积概率）最大。搜索的算法是一种动态规划剪枝的算法，称之为 Viterbi 算法，用于寻找全局最优路径。感兴趣的同学可以到 Wikipedia 上搜一下。



这里所说的累积概率，由三部分构成，分别是：

4 观察概率：每帧和每个状态对应的概率
5 转移概率：每个状态转移到自身或转移到下个状态的概率
6 语言概率：根据语言统计规律得到的概率

其中，前两种概率从声学模型中获取，最后一种概率从语言模型中获取。语言模型是使用大量的文本训练出来的，存储的是任意单词、任意两个单词、任意三个单词（通常也就到三个单词）在大量文本中的出现机率。

这样基本上语音识别过程就完成了。

以上介绍的是传统的基于 HMM 的语音识别。以上的文字不追求严谨，只是想让大家容易理解。

如果感兴趣，想进一步了解，HTK Book 是非常好的入门书，这本书实际上是剑桥大学发布的著名开源工具包 HTK Speech Recognition Toolkit 的说明书，近400页，厚厚的一本。如果有时间、有兴趣，可以照着书中的第二章在电脑上做一遍，你将搭建出一个简单但基本完整的语音识别系统，能识别简单的英语数字串。

2.2 tornadomeet 的 cnblog

网址： http://www.cnblogs.com/tornadomeet/archive/2013/08/23/3276753.html

为了对 GMM-HMM 在语音识别上的应用有个宏观认识，花了些时间读了下 HTK（用 htk 完成简单的孤立词识别）的部分源码，对该算法总算有了点大概认识，达到了预期我想要的。不得不说，网络上关于语音识别的通俗易懂教程太少，都是各种公式满天飞，很少有说具体细节的，当然了，那需要有实战经验才行。下面总结以下几点，对其有个宏观印象即可（以孤立词识别为例）。

一、每个单词的读音都对应一个 HMM 模型，大家都知道 HMM 模型中有个状态集 S，那么每个

状态用什么来表示呢，数字？向量？矩阵？其实这个状态集中的状态没有具体的数学要求，只是一个名称而已，你可以用'1', '2', '3'…表示，也可以用'a', 'b', 'c '表示。另外每个 HMM 模型中到底该用多少个状态，是通过先验知识人为设定的。

二、HMM 的每一个状态都对应有一个观察值，这个观察值可以是一个实数，也可以是个向量，且每个状态对应的观察值的维度应该相同。假设现在有一个单词的音频文件，首先需要将其进行采样得到数字信息（A/D 转换），然后分帧进行 MFCC 特征提取，假设每一帧音频对应的 MFCC 特征长度为39，则每个音频文件就转换成了 N 个 MFCC 向量（不同音频文件对应的 N 可能不同），这就成了一个序列，而在训练 HMM 模型的参数时（比如用 Baum-Welch 算法），每次输入到 HMM 中的数据要求就是一个观测值序列。这时，每个状态对应的观测值为39维的向量，因为向量中元素的取值是连续的，需要用多维密度函数来模拟，通常情况下用的是多维高斯函数。在 GMM-HMM 体系中，这个拟合函数是用 K 个多维高斯混合得到的。假设知道了每个状态对应的 K 个多维高斯的所有参数，则该 GMM 生成该状态上某一个观察向量（一帧音频的 MFCC 系数）的概率就可以求出来了。

三、对每个单词建立一个 HMM 模型，需要用到该单词的训练样本，这些训练样本是提前标注好的，即每个样本对应一段音频，该音频只包含这个单词的读音。当有了该单词的多个训练样本后，就用这些样本结合 Baum-Welch 算法和 EM 算法来训练出 GMM-HMM 的所有参数，这些参数包括初始状态的概率向量，状态之间的转移矩阵，每个状态对应的观察矩阵（这里对应的是 GMM，即每个状态对应的 K 个高斯的权值，每个高斯的均值向量和方差矩阵）。

四、在识别阶段，输入一段音频，如果该音频含有多个单词，则可以手动先将其分割开（考虑的是最简单的方法），然后提取每个单词的音频 MFCC 特征序列，将该序列输入到每个 HMM 模型（已提前训练好的）中，采用前向算法求出每个 HMM 模型生成该序列的概率，最后取最大概率对应的那个模型，而那个模型所表示的单词就是我们识别的结果。

五、在建立声学模型时，可以用 Deep Learning 的方法来代替 GMM-HMM 中的 GMM，因为 GMM 模拟任意函数的功能取决于混合高斯函数的个数，所以具有一定的局限性，属于浅层模型。而 Deep Network 可以模拟任意的函数，因而表达能力更强。注意，这里用来代替 GMM 的 Deep Nets 模型要求是产生式模型，比如 DBN，DBM 等，因为在训练 HMM-DL 网络时，需要用到 HMM 的某个状态产生一个样本的概率。

六、GMM-HMM 在具体实现起来还是相当复杂的。

七、一般涉及到时间序列时才会使用 HMM，比如这里音频中的语音识别，视频中的行为识别等。如果我们用 GMM-HMM 对静态的图片分类，因为这里没涉及到时间信息，所以 HMM 的状态数可设为1，那么此时的 GMM-HMM 算法就退化成 GMM 算法了。

2.3 zouxy09 的博客

网址：http://blog.csdn.net/zouxy09/article/details/7941585

语音识别技术就是让机器通过识别和理解过程把语音信号转变为相应的文本或命令的技术。

基于语音识别芯片的嵌入式产品也越来越多，如 Sensory 公司的 RSC 系列语音识别芯片、Infineon 公司的 Unispeech 和 Unilite 语音芯片等，这些芯片在嵌入式硬件开发中得到了广泛的应用。在软件上，目前比较成功的语音识别软件有：Nuance、IBM 的 Viavoice 和 Microsoft 的

SAPI 以及开源软件 HTK，这些软件都是面向非特定人、大词汇量的连续语音识别系统。

语音识别本质上是一种模式识别的过程，未知语音的模式与已知语音的参考模式逐一进行比较，最佳匹配的参考模式被作为识别结果。

语音识别的目的就是让机器赋予人的听觉特性，听懂人说什么，并作出相应的动作。目前大多数语音识别技术是基于统计模式的，从语音产生机理来看，语音识别可以分为语音层和语言层两部分。

当今语音识别技术的主流算法，主要有基于动态时间规整(DTW)算法、基于非参数模型的矢量量化(VQ)方法、基于参数模型的隐马尔可夫模型(HMM)的方法、基于人工神经网络(ANN)和支持向量机等语音识别方法。

语音识别分类：
根据对说话人的依赖程度，分为：
（1）特定人语音识别（SD）：只能辨认特定使用者的语音，训练→使用。
（2）非特定人语音识别（SI）：可辨认任何人的语音，无须训练。
根据对说话方式的要求，分为：
（1）孤立词识别：每次只能识别单个词汇。
（2）连续语音识别：用者以正常语速说话，即可识别其中的语句。

语音识别系统的模型通常由声学模型和语言模型两部分组成，分别对应于语音到音节概率的计算和音节到字概率的计算。

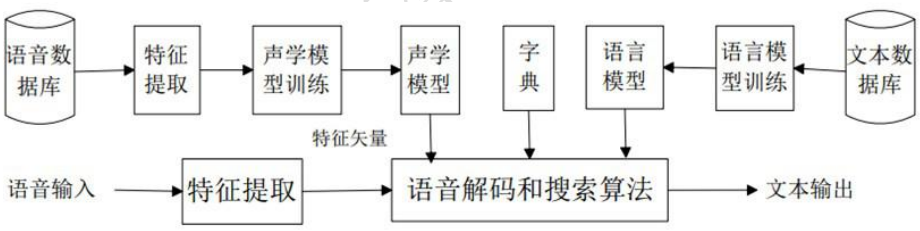


图2-1 连续语音识别框图

Sphinx 是由美国卡内基梅隆大学开发的大词汇量、非特定人、连续英语语音识别系统。一个连续语音识别系统大致可分为四个部分：特征提取，声学模型训练，语言模型训练和解码器。
（1）预处理模块：

对输入的原始语音信号进行处理，滤除掉其中的不重要的信息以及背景噪声，并进行语音信号的端点检测（找出语音信号的始末）、语音分帧（近似认为在10-30ms 内是语音信号是短时平稳的，将语音信号分割为一段一段进行分析）以及预加重（提升高频部分）等处理。
（2）特征提取：

去除语音信号中对于语音识别无用的冗余信息，保留能够反映语音本质特征的信息，并用一定的形式表示出来。也就是提取出反映语

音信号特征的关键特征参数形成特征矢量序列，以便用于后续处理。

目前的较常用的提取特征的方法还是比较多的，不过这些提取方法都是由频谱衍生出来的。Mel 频率倒谱系数（MFCC)参数因其良好的抗噪性和鲁棒性而应用广泛。在 sphinx 中也是用 MFCC 特征的。MFCC 的计算首先用 FFT 将时域信号转化成频域，之后对其对数能量谱用依照 Mel 刻度分布的三角滤波器组进行卷积，最后对各个滤波器的输出构成的向量进行离散余弦变换 DCT，取前 N 个系数。

在 sphinx 中，用帧 frames 去分割语音波形，每帧大概10ms，然后每帧提取可以代表该帧语音的39个数字，这39个数字也就是该帧语音的 MFCC 特征，用特征向量来表示。

（3）声学模型训练：

根据训练语音库的特征参数训练出声学模型参数。在识别时可以将待识别的语音的特征参数同声学模型进行匹配，得到识别结果。

目前的主流语音识别系统多采用隐马尔可夫模型 HMM 进行声学模型建模。声学模型的建模单元，可以是音素，音节，词等各个层次。对于小词汇量的语音识别系统，可以直接采用音节进行建模。而对于词汇量偏大的识别系统，一般选取音素，即声母，韵母进行建模。识别规模越大，识别单元选取的越小。（关于 HMM，网上有很多经典的解说，例如《HMM 学习最佳范例》和《隐马尔科夫模型(hmm)简介》等，不了解的可以去看看）

HMM 是对语音信号的时间序列结构建立统计模型，将其看作一个数学上的双重随机过程:一个是用具有有限状态数的 Markov 链来模拟语音信号统计特性变化的隐含（马尔可夫模型的内部状态外界不可见）的随机过程，另一个是与 Markov 链的每一个状态相关联的外界可见的观测序列（通常就是从各个帧计算而得的声学特征）的随机过程。

人的言语过程实际上就是一个双重随机过程，语音信号本身是一个可观测的时变序列，是由大脑根据语法知识和言语需要（不可观测的状态）发出的音素的参数流（发出的声音）。HMM 合理地模仿了这一过程，是较为理想的一种语音模型。用 HMM 刻画语音信号需作出两个假设，一是内部状态的转移只与上一状态有关，另一是输出值只与当前状态（或当前的状态转移）有关，这两个假设大大降低了模型的复杂度。

语音识别中使用 HMM 通常是用从左向右单向、带自环、带跨越的拓扑结构来对识别基元建模，一个音素就是一个三至五状态的 HMM，一个词就是构成词的多个音素的 HMM 串行起来构成的 HMM，而连续语音识别的整个模型就是词和静音组合起来的 HMM。

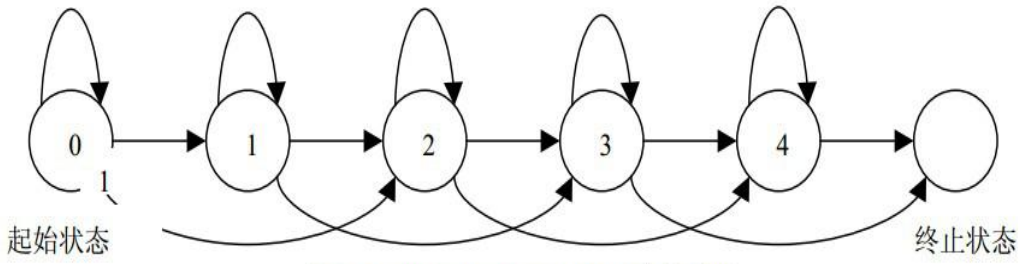


图 2-3：PocketSphinx HMM 状态转移图

（4）语言模型训练：

语言模型是用来计算一个句子出现概率的概率模型。它主要用于决定哪个词序列的可能性更大，或者在出现了几个词的情况下预测下一个即将出现的词语的内容。换一个说法说，语言模型是用来约束单词搜索的。它定义了哪些词能跟在上一个已经识别的词的后面（匹配是一个顺序的处理过程），这样就可以为匹配过程排除一些不可能的单词。

语言建模能够有效的结合汉语语法和语义的知识，描述词之间的内在关系，从而提高识别率，减少搜索范围。语言模型分为三个层次：字典知识，语法知识，句法知识。

对训练文本数据库进行语法、语义分析，经过基于统计模型训练得到语言模型。语言建模方法主要有基于规则模型和基于统计模型两种方法。统计语言模型是用概率统计的方法来揭示语言单位内在的统计规律，其中 N-Gram 模型简单有效，被广泛使用。它包含了单词序列的统计。

N-Gram 模型基于这样一种假设，第 n 个词的出现只与前面 N-1个词相关，而与其它任何词都不相关，整句的概率就是各个词出现概率的乘积。这些概率可以通过直接从语料中统计 N 个词同时出现的次数得到。常用的是二元的 Bi-Gram 和三元的 Tri-Gram。

Sphinx 中是采用二元语法和三元语法的统计语言概率模型，也就是通过前一个或两个单词来判定当前单词出现的概率 P(w2| w1)，P(w3| w2, w1)。

（5）语音解码和搜索算法：

解码器：即指语音技术中的识别过程。针对输入的语音信号，根据已经训练好的HMM 声学模型、语言模型及字典建立一个识别网络，根据搜索算法在该网络中寻找最佳的一条路径，这个路径就是能够以最大概率输出该语音信号的词串，这样就确定这个语音样本所包含的文字了。所以解码操作即指搜索算法：是指在解码端通过搜索技术寻找最优词串的方法。

连续语音识别中的搜索，就是寻找一个词模型序列以描述输入语音信号，从而得到词解码序列。搜索所依据的是对公式中的声学模型打分和语言模型打分。在实际使用中，往往要依据经验给语言模型加上一个高权重，并设置一个长词惩罚分数。当今的主流解码技术都是基于 Viterbi 搜索算法的，Sphinx 也是。

基于动态规划的 Viterbi 算法在每个时间点上的各个状态，计算解码状态序列对观察序列的后验概率，保留概率最大的路径，并在每个节点记录下相应的状态信息以便最后反向获取词解码序列。Viterbi 算法本质上是一种动态规划算法，该算法遍历 HMM 状态网络并保留每一帧语音在某个状态的最优路径得分。

连续语音识别系统的识别结果是一个词序列。解码实际上是对词表的所有词反复搜索。词表中词的排列方式会影响搜索的速度，而词的排列方式就是字典的表示形式。Sphinx 系统中采用音素作为声学训练单元，通常字典就用来记录每个单词由哪些个音素组成，也可以理解为对每个词的发音进行标注。

N-best 搜索和多遍搜索：为在搜索中利用各种知识源，通常要进行多遍搜索，第一遍使用代价低的知识源（如声学模型、语言模型和音标词典），产生一个候选列表或词候选网格，在此基础上进行使用代价高的知识源（如4阶或5阶的 N-Gram、4阶或更高的上下文相关模型）的第二遍搜索得到最佳路径。

对于语音识别过程个人的理解：

例如我对电脑说：“帮我打开“我的电脑”！”然后电脑得理解我说了什么，然后再执行打开“我的电脑”的操作，那怎么实现呢？

这个得预先有一个工作，就是电脑得先学会“帮我打开“我的电脑”！”这句语音（实际上是一个波形）所代表的文字就是“帮我打开“我的电脑”！”这句词串。那么如何让它学会呢？

如果以音节（对汉语来说就是一个字的发音）为语音基元的话，那么电脑就是一个字一个字地学习，例如“帮”字、“我”字等等，那么“帮”字怎么学习呢？也就是说电脑接收到一个“帮”字的语音波形，怎么分析理解才知道它代表的是“帮”字呢？首先我们需要建立一个数学模型来表示这个语音。因为语音是连续的不平稳的信号，但是在短的时间内可以认为是平稳的，所以我们需要分割语音信号为一帧一帧，假如大概 25ms 一帧，然后为了让每一帧平稳过渡，我们就让每帧见存在重叠，假如重叠10ms。这样每帧的语言信号就是平稳的了，再从每帧语音信号中提取反映语音本质特征的信息（去除语音信号中对于语音识别无用的冗余信息，同时达到降维）。那么采用什么特征最能表达每一帧的语音呢？MFCC 是用的比较多的一种，这里不介绍了。然后我们就提取每一帧语音的 MFCC 特征，得到了是一系列的系数，大概四五十个这样，sphinx 中是39个数字，组成了特征向量。好，那么我们就通过39个数字来描述每一帧的语音了，那不同的语音帧就会有不同的39个数字的组合，那我们用什么数学模型去描述这39个数字的分布情况呢？这里我们可以用一个混合高斯模型来表示着39个数字的分布，而混合高斯模型就存在着两个参数：均值和方差；那么实际上每一帧的语音就对应着这么一组均值和方差的参数了。呵呵，挺啰嗦的啊。

好了，这样“帮”字的语音波形中的一帧就对应了一组均值和方差（HMM 模型中的观察序列），那么我们只需要确定“帮”字（HMM 模型中的隐含序列）也对应于这一组均值和方差就可以了。那么后者是怎

么对应的呢？这就是训练的作用了！我们知道描述一个 HMM 模型需要三个参数：初始状态概率分布π、隐含状态序列的转移矩阵 A（就是某个状态转移到另一个状态的概率观察序列中的这个均值或者方差的概率）和某个隐含状态下输出观察值的概率分布 B（也就是某个隐含状态下对应于）；而声学模型可以用 HMM 模型来建模，也就是对于每一个建模的语音单元，我们需要找到一组 HMM 模型参数（π，A，B）就可以代表这个语音单元了。那么这三个参数怎么确定呢？训练！我们给出一个语音的数据库，指明说这个语音代表这个词，然后让电脑去学习，也就是对数据库进行统计，得到（π，A，B）这三个参数。

好了，一个字（建模单元）的声学模型建立了。那汉语是不是有很多个字啊，那我们就得对每一个建立声学模型了。假设就有几千个模型，然后每个模型就有了三个或者5个 HMM 状态，那么如果你说的句子有10个字，那我们就得搜索这所有可能的模型去匹配你的语音，那是多么大的搜索空间了，这非常耗时。那我们就需要采用一个比较优的搜索算法了（这里是 Viterbi-Beam 算法），它每搜索到一个状态点，就保留概率最大的，然后舍弃之前的状态点，这样就裁剪了很多的搜索路径，但因为忽略了之前的路径，所以它就只能得到一个局部的最优解。

那假如出现以下情况呢？例如，It's a nice day，从语音上可能会被识别为：It sun niced A，或者是 It son ice day。从声学模型来看它是无法区别这些结果，因为其不同之处只是在于每个单词的边界划分位置不同造成的。这时候语言模型就该闪亮登场了，从语义上判断那个结果出现的概率最大，即为搜索结果。语言模型 N-Gram 基于这样一种假设，第 n 个词的出现只与前面 N-1个词相关，而与其它任何词都不相关，整句的概率就是各个词出现概率的乘积。这些概率可以通过直接从语料中统计 N 个词同时出现的次数得到。这样就可以约束搜索，增加识别的准确率了。

好了，有点啰嗦啊，我也是前几天为了要将语音识别加入到我的人机交互系统，然后才去了解语音识别的，所以可能理解不是很正确，希望各位不吝指正！谢谢

对于一些语音的基本概念我翻译了 CMUsphinx 的 wiki 的部分，具体见：http://blog.csdn.net/zouxy09/article/details/7941055

补充的一些概念：（整理自百度百科）

音节：

音节是听觉能感受到的最自然的语音单位，有一个或几个音素按一定规律组合而成。

汉语音节:

汉语中一个汉字就是一个音节，每个音节由声母、韵母和声调三个部分组成；汉语普通话中的无调音节（不做音调区分）共有400个音节。拼音是拼读音节的过程，就是按照普通话音节的构成规律，把声母、韵母、声调急速连续拼合并加上声调而成为一个音节。如：

q-i-áng→qiáng（强）。

英语音节：

音节是读音的基本单位，任何单词的读音，都是分解为一个个音节朗读。英语中一个元音音素可构成一个音节，一个元音音素和一个或几个辅音音素结合也可以构成一个音节。英语的词有一个音节的，两个音节的，多个音节的。一个音节叫单音节词，两个音节叫双音节词，三个音节以上叫多音节。如：take 拿，ta'ble 桌子，po‘ta'to 马铃薯，po’pu‘la'tion 人口，con’gra‘tu’la'tion 祝贺。te‘le’com‘mu‘ni’ca'tion 电讯。

元音音素是构成音节的主体，辅音是音节的分界线。每个元音音素都可以构成一个音节，如：bed 床，bet 打赌。两个元音音素都可以构成一个音节，如：seat 坐位，beat 毒打，beast 极好的。两元音音素之间有一个辅音音素时，辅音音素归后一音节，如：stu'dent 学生，la'bour 劳动。有两个辅音音素时，一个辅音音素归前一音节，一个归后一音节，如： win'ter 冬天 fa'ther 父亲，tea'cher 教师。

音素：

音素是根据语音的自然属性划分出来的最小语音单位。从声学性质来看，音素是从音质角度划分出来的最小语音单位。从生理性质来看，一个发音动作形成一个音素。如〔ma〕包含〔m〕〔a〕两个发音动作，是两个音素。相同发音动作发出的音就是同一音素，不同发音动作发出的音就是不同音素。如〔ma-mi〕中，两个〔m〕发音动作相同，是相同音素，〔a〕〔i〕发音动作不同，是不同音素。

汉语音素：

音节只是最自然的语音单位，而音素是最小的语音单位音素。汉语包括10个元音，22个辅音，总共有32个。一个音节，至少有一个音素，至多有四个音素。如“普通话”，由三个音节组成（每个字一个音节），可以分析成“p,u,t,o,ng,h,u,a”八个音素。

英语音素：

记录英语音素的符号叫做音标。英语国际音标共有48个音素，其中元音音素20个，辅音音素28个。英语辅音和元音在语言中的作用，就相当于汉语中的声母和韵母。

语料：

通常，在统计自然语言处理中实际上不可能观测到大规模的语言实例。所以，人们简单地用文本作为替代，并把文本中的上下文关系作为现实世界中语言的上下文关系的替代品。我们把一个文本集合称为语料库（Corpus），当有几个这样的文本集合的时候，我们称之为语料库集合（Corpora）。语料库通常指为语言研究收集的、用电子形式保存的语言材料，由自然出现的书面语或口语的样本汇集而成，用来代表特定的语言或语言变体。

语料库就是把平常我们说话的时候的句子、一些文学作品的语句段落、报刊杂志上出现过的语句段落等等在现实生活中真实出现过的语言材料整理在一起，形成一个语料库，以便做科学研究的时候能够从中取材或者得到数据佐证。
例如我如果想写一篇关于“给力”这个词的普及性的文章，就可以到语料库中查询这个词出现的频率、用法等等。

Reference
王韵，基于 Sphinx 的汉语连续语音识别，太原理工大学，硕士学位论文

2.4 美女 zhang 的博客

网址：http://blog.csdn.net/abcjennifer/article/details/27346787

GMM-HMM 语音识别模型 原理篇

本文简明讲述 GMM-HMM 在语音识别上的原理，建模和测试过程。这篇 blog 只回答三个问题：

1. 什么是 Hidden Markov Model?
    HMM 要解决的三个问题:
    1) Likelihood
    2) Decoding
    3) Training
2. GMM 是神马？怎样用 GMM 求某一音素（phoneme）的概率？
3. GMM+HMM 大法解决语音识别
    3.1 识别
    3.2 训练
        3.2.1 Training the params of GMM
        3.2.2 Training the params of HMM

首先声明我是做视觉的不是做语音的，迫于**需要 24 小时速成语音。上网查 GMM-HMM 资料中文几乎为零，英文也大多是 paper。苦苦追寻终于貌似搞懂了 GMM-HMM，感谢语音组老夏（http://weibo.com/ibillxia）提供资料给予指导。本文结合最简明的概括还有自己一些理解应运而生，如有错误望批评指正。

==========

1. 什么是 Hidden Markov Model?

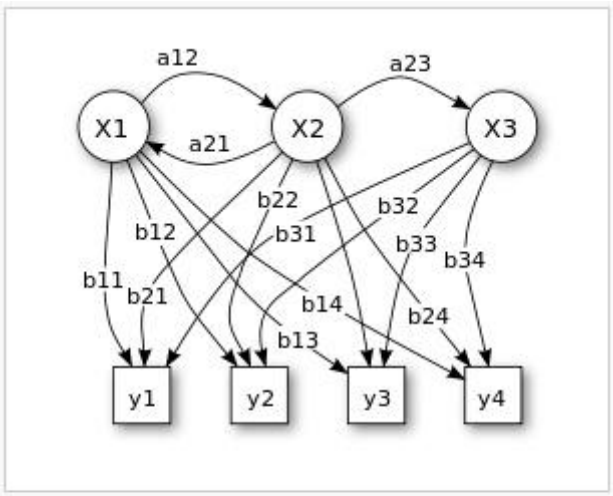


Figure 1. Probabilistic parameters of a hidden Markov model (example)
x — states
y — possible observations
a — state transition probabilities
b — output probabilities

ANS：一个有隐节点（unobservable）和可见节点（visible）的马尔科夫过程（见详解）。隐节点表示状态，可见节点表示我们听到的语音或者看到的时序信号。
最开始时，我们指定这个 HMM 的结构，训练 HMM 模型时：给定 n 个时序信号 y1...yT（训练样本），用 MLE（typically implemented in EM） 估计参数：
1. N 个状态的初始概率
2. 状态转移概率 a
3. 输出概率 b

- 在语音处理中，一个 word 由若干 phoneme（音素）组成；
- 每个 HMM 对应于一个 word 或者音素（phoneme）
- 一个 word 表示成若干 states，每个 state 表示为一个音素

用 HMM 需要解决 3 个问题：
1）. Likelihood: 一个 HMM 生成一串 observation 序列 x 的概率< the Forward algorithm>

- Initialization

$$\alpha_0(s_I) = 1$$
$$\alpha_0(s_j) = 0 \qquad \text{if } s_j \neq s_I$$

- Recursion

$$\alpha_t(s_j) = \sum_{i=1}^{N} \alpha_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$

- Termination

$$p(\mathbf{X} \mid \boldsymbol{\lambda}) = \alpha_T(s_E) = \sum_{i=1}^{N} \alpha_T(s_i) a_{iE}$$

http://blog.csdn.net/a[illegible]jennifer

其中，αt（sj）表示 HMM 在时刻 t 处于状态 j，且 observation = {x1,...,xt} 的概率 $\alpha_t(s_j) = p(\mathbf{x}_1, \ldots, \mathbf{x}_t, S(t) = s_j \mid \boldsymbol{\lambda})$ ，

aij 是状态 i 到状态 j 的转移概率，bj（xt）表示在状态 j 的时候生成 xt 的概率，

2）. Decoding: 给定一串 observation 序列 x，找出最可能从属的 HMM 状态序列< the Viterbi algorithm>

在实际计算中会做剪枝，不是计算每个可能 state 序列的 probability，而是用 Viterbi approximation：

从时刻 1：t，只记录转移概率最大的 state 和概率。

记 Vt(si) 为从时刻 t-1 的所有状态转移到时刻 t 时状态为 j 的最大概率：

$$V_t(s_j) = \max_i V_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$

记 $bt_t(s_i)$ 为：从时刻 t-1 的哪个状态转移到时刻 t 时状态为 j 的概率最大；

进行 Viterbi approximation 过程如下：

- Initialization

$$V_0(s_I) = 1$$
$$V_0(s_j) = 0 \qquad \text{if } s_j \neq s_I$$
$$bt_0(s_j) = 0$$

- Recursion

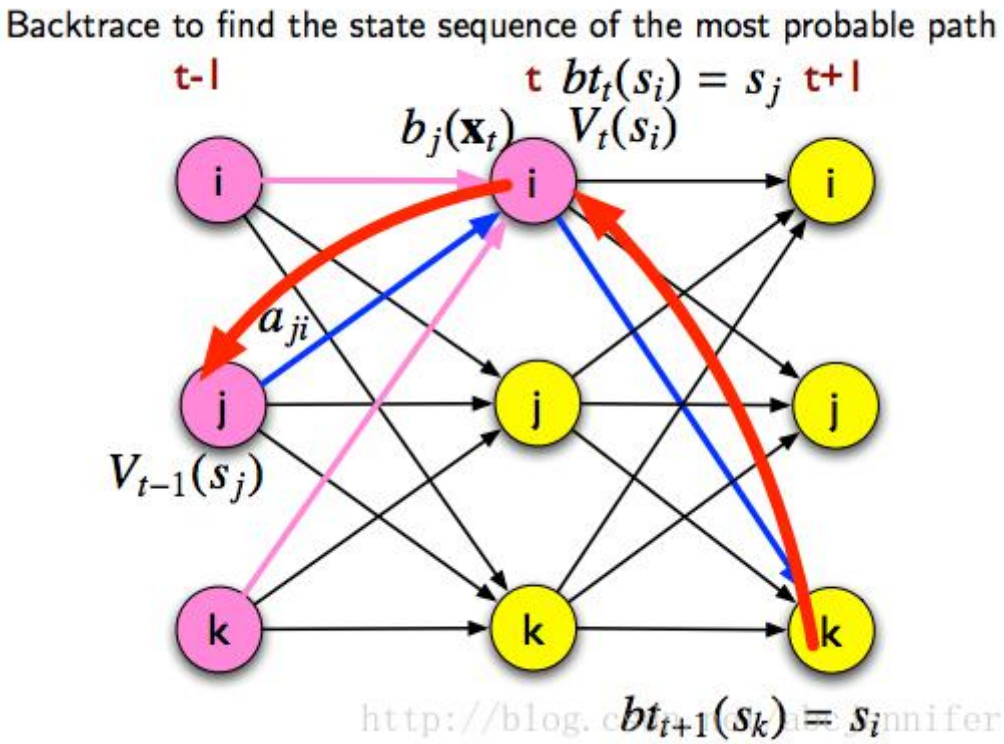
$$V_t(s_j) = \max_{i=1}^{N} V_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$
$$bt_t(s_j) = \arg\max_{i=1}^{N} V_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$

- Termination

$$P^* = V_T(s_E) = \max_{i=1}^{N} V_T(s_i) a_{iE}$$
$$s_T^* = bt_T(q_E) = \arg\max_{i=1}^{N} V_T(s_i) a_{iE}$$

http://blog.csdn.net/abcjennifer

然后根据记录的最可能转移状态序列 $bt_t(s_i)$ 进行回溯：



http://blog.csdn.net/abcjennifer

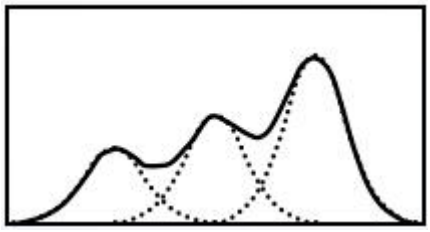
3）. Training: 给定一个 observation 序列 x，训练出 HMM 参数 λ = {aij, bij} the EM (Forward-Backward) algorithm

这部分我们放到“3. GMM+HMM 大法解决语音识别”中和 GMM 的 training 一起讲

---

2. GMM 是神马？怎样用 GMM 求某一音素（phoneme）的概率？

2.1 简单理解混合高斯模型就是几个高斯的叠加。。。e.g. k=3



$$p(\mathbf{x}) = \sum_{j=1}^{P} P(j)p(\mathbf{x}|j) = \sum_{j=1}^{P} P(j)N_j(\mathbf{x}; \mu_j, \sigma_j^2)$$

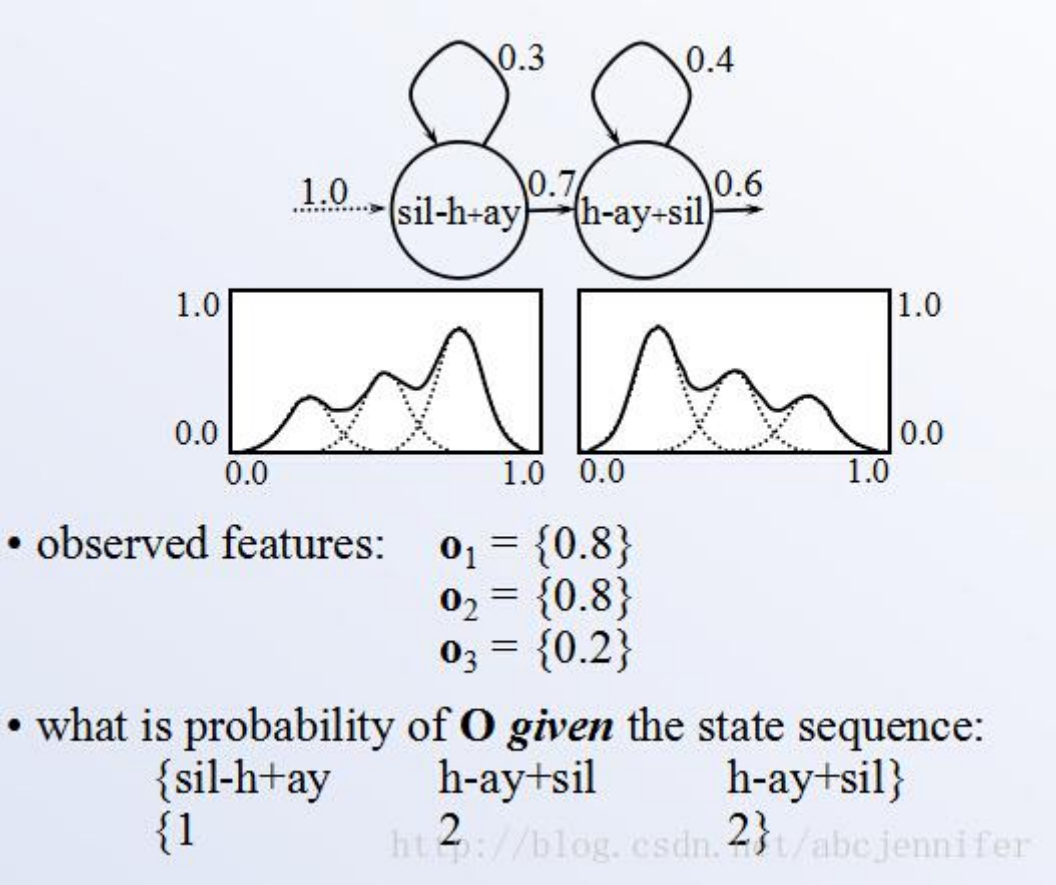
http://blog.csdn.net/abcjennifer

fig2. GMM illustration and the probability of x

2.2 GMM for state sequence

每个 state 有一个 GMM，包含 k 个高斯模型参数。如”hi“（k=3）：

PS：sil 表示 silence（静音）



http://blog.csdn.net/abcjennifer

fig3. use GMM to estimate the probability of a state sequence given observation {o1, o2, o3}

其中，每个 GMM 有一些参数，就是我们要 train 的输出概率参数

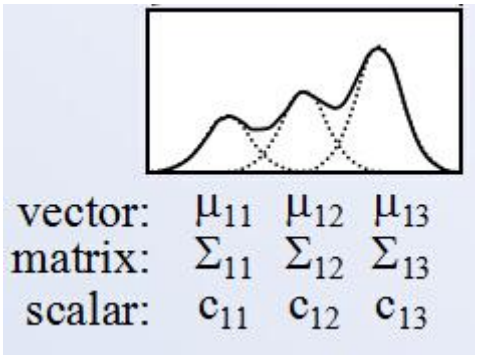


fig4. parameters of a GMM

怎么求呢？和 KMeans 类似，如果已知每个点 x^n 属于某每类 j 的概率 p(j|x^n)，则可以估计其参数:

$$\hat{\mu}_j = \frac{\sum_n P(j|\mathbf{x}^n)\mathbf{x}^n}{\sum_n P(j|\mathbf{x}^n)} = \frac{\sum_n P(j|\mathbf{x}^n)\mathbf{x}^n}{N_j^*}$$

$$\hat{\sigma}_j^2 = \frac{\sum_n P(j|\mathbf{x}^n)||\mathbf{x}^n - \boldsymbol{\mu}_k||^2}{\sum_n P(j|\mathbf{x}^n)} = \frac{\sum_n P(j|\mathbf{x}^n)||\mathbf{x}^n - \boldsymbol{\mu}_k||^2}{N_j^*}$$

$$\hat{P}(j) = \frac{1}{N}\sum_n P(j|\mathbf{x}^n) = \frac{N_j^*}{N}$$

http://blog.csdn.net/abcjennifer

其中

$$N_j^* = \sum_{n=1}^{N} P(j|\mathbf{x}^n)$$

只要已知了这些参数，我们就可以在 predict（识别）时在给定 input sequence 的情况下，计算出一串状态转移的概率。如上图要计算的 state sequence 1->2->2 概率：

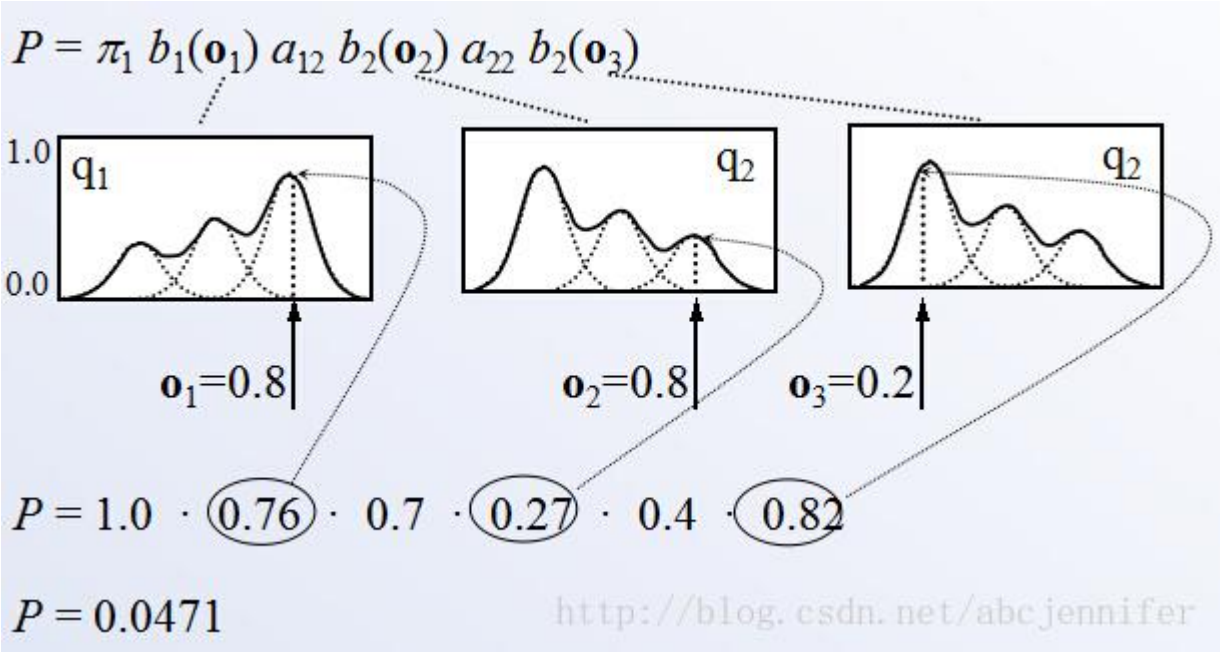


fig5. probability of S1->S2->S3 given o1->o2->o3

---

3. GMM+HMM 大法解决语音识别

<!--识别-->

我们获得 observation 是语音 waveform，以下是一个词识别全过程：

1). 将 waveform 切成等长 frames，对每个 frame 提取特征（e.g. MFCC），

2).对每个 frame 的特征跑 GMM，得到每个 frame(o_i)属于每个状态的概率 b_state(o_i)

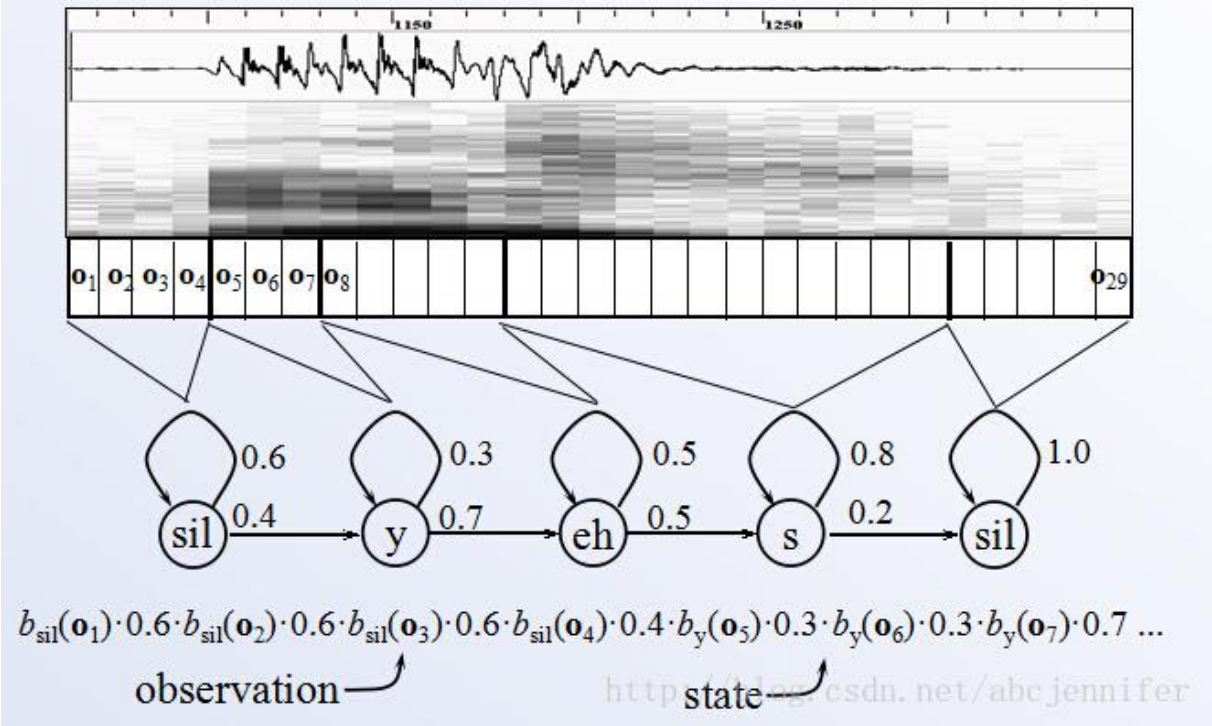


fig6. complete process from speech frames to a state sequence

3). 根据每个单词的 HMM 状态转移概率 a 计算每个状态 sequence 生成该 frame 的概率; 哪个词的 HMM 序列跑出来概率最大，就判断这段语音属于该词

宏观图：

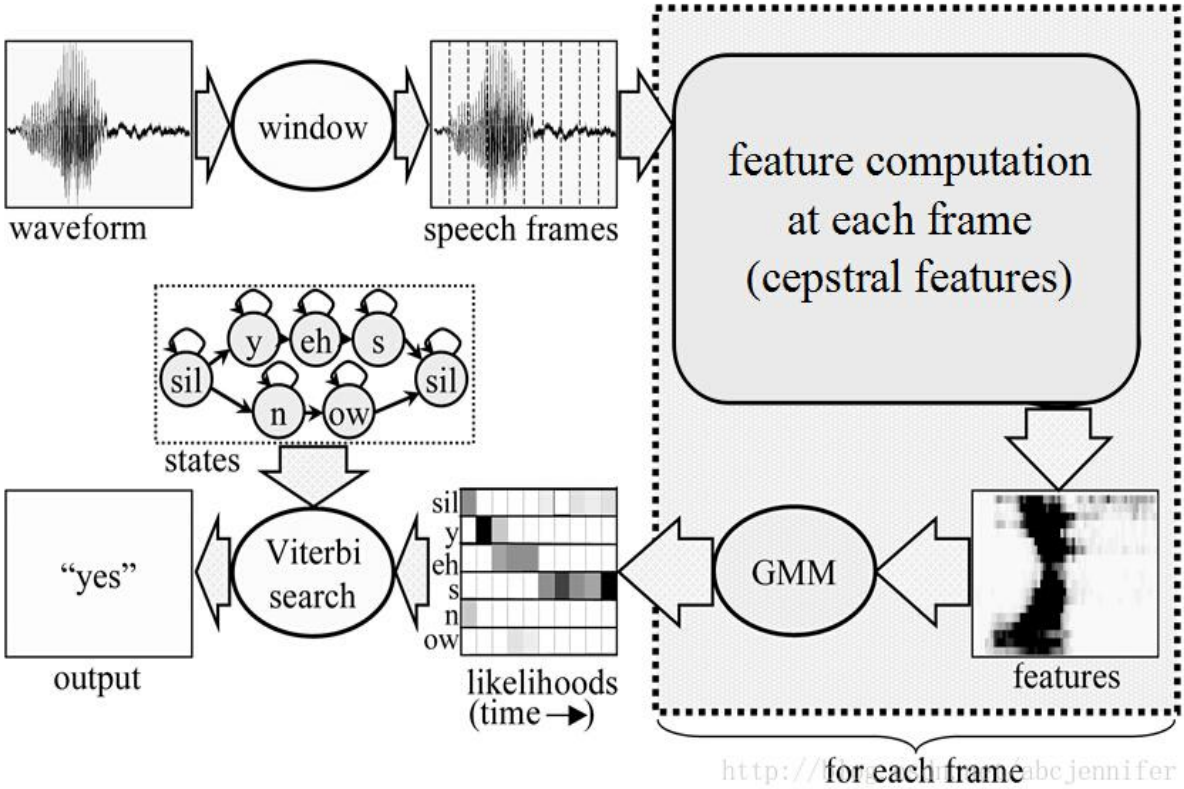


fig7. Speech recognition, a big framework
(from Encyclopedia of Information Systems, 2002)

<!--训练-->

好了，上面说了怎么做识别。那么我们怎样训练这个模型以得到每个 GMM 的参数和 HMM 的转移概率什么的呢？

①Training the params of GMM

GMM 参数：高斯分布参数：mean vector μ^j; covariance matrix Σ^j

从上面 fig4 下面的公式我们已经可以看出来想求参数必须要知道 P(j|x),即，x 属于第 j 个高斯的概率。怎么求捏？

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

fig8. bayesian formula of P(j | x)

根据上图 P（j | x），我们需要求 P(x|j)和 P（j）去估计 P(j|x).

这里由于 P(x|j)和 P（j）都不知道，需要用 EM 算法迭代估计以最大化 P(x) = P(x1)*p(x2)*...*P(xn)：

A. 初始化（可以用 kmeans）得到 P(j)

B. 迭代

E（estimate）-step: 根据当前参数 (means, variances, mixing parameters)估计 P(j|x)

M（maximization）-step: 根据当前 P(j|x) 计算 GMM 参数（根据 fig4 下面的公式：）

$$\hat{\mu}_j = \frac{\sum_n P(j|\mathbf{x}^n)\mathbf{x}^n}{\sum_n P(j|\mathbf{x}^n)} = \frac{\sum_n P(j|\mathbf{x}^n)\mathbf{x}^n}{N_j^*}$$

$$\hat{\sigma}_j^2 = \frac{\sum_n P(j|\mathbf{x}^n)||\mathbf{x}^n - \mu_k||^2}{\sum_n P(j|\mathbf{x}^n)} = \frac{\sum_n P(j|\mathbf{x}^n)||\mathbf{x}^n - \mu_k||^2}{N_j^*}$$

$$\hat{P}(j) = \frac{1}{N}\sum_n P(j|\mathbf{x}^n) = \frac{N_j^*}{N}$$

http://blog.csdn.net/abcjennifer

，

其中 $$N_j^* = \sum_{n=1}^{N} P(j|\mathbf{x}^n)$$

②Training the params of HMM

前面已经有了 GMM 的 training 过程。在这一步，我们的目标是：从 observation 序列中估计 HMM 参数 λ；

假设状态->observation 服从单核高斯概率分布：$$b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j) = \mathcal{N}(\mathbf{x}; \mu^j, \Sigma^j)$$，

则 λ 由两部分组成：

Parameters $\boldsymbol{\lambda}$:

- Transition probabilities a_{ij}:

$$\sum_j a_{ij} = 1$$

- Gaussian parameters for state s_j:
  mean vector μ^j; covariance matrix Σ^j

http://blog.csdn.net/abcjennifer

HMM 训练过程：迭代

E（estimate）-step: 给定 observation 序列，估计时刻 t 处于状态 sj 的概率 $\gamma_t(s_j)$

M（maximization）-step: 根据$\gamma_t(s_j)$重新估计 HMM 参数 aij.

其中，

E-step: 给定 observation 序列，估计时刻 t 处于状态 sj 的概率 $\gamma_t(s_j)$

为了估计$\gamma_t(s_j)$，定义 $\beta_t(s_j)$: t 时刻处于状态 sj 的话，t 时刻未来 observation 的概率。即 $\beta_t(s_j) = p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \mathbf{x}_T \mid S(t) = s_j, \boldsymbol{\lambda})$

这个可以递归计算：β_t（si）=从状态 si 转移到其他状态 sj 的概率 aij * 状态 i 下观测到 x_{t+1}的概率 bi(x_{t+1}) * t 时刻处于状态 sj 的话{t+1}后 observation 概率β_{t+1}（sj）

即：

- Initialisation

$$\beta_T(s_i) = a_{iE}$$

- Recursion

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

- Termination

$$p(\mathbf{X} \mid \boldsymbol{\lambda}) = \beta_0(s_I) = \sum_{j=1}^{N} a_{Ij} b_j(\mathbf{x}_1) \beta_1(s_j) = \alpha_T(s_E)$$

http://blog.csdn.net/abcjennifer

定义刚才的$\gamma_t(s_j)$为 state occupation probability，表示给定 observation 序列，时刻 t 处于状态 sj 的概率 P(S(t)=sj | X, λ)。根据贝叶斯公式 p(A|B,C) = P(A,B|C)/P(B|C)，有：

$$P(S(t) = s_j \mid \mathbf{X}, \boldsymbol{\lambda}) = \frac{p(\mathbf{X}, S(t) = s_j \mid \boldsymbol{\lambda})}{p(\mathbf{X} \mid \boldsymbol{\lambda})}$$

http://blog.csdn.net/abcjennifer

由于分子 p(A,B|C)为

$$\begin{aligned}\alpha_t(s_j)\beta_t(s_j) &= p(\mathbf{x}_1, \ldots, \mathbf{x}_t, S(t) = s_j \mid \boldsymbol{\lambda}) \\ &\quad p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \mathbf{x}_T \mid S(t) = s_j, \boldsymbol{\lambda}) \\ &= p(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots, \mathbf{x}_T, S(t) = s_j \mid \boldsymbol{\lambda}) \\ &= p(\mathbf{X}, S(t) = s_j \mid \boldsymbol{\lambda})\end{aligned}$$

http://blog.csdn.net/abcjennifer

其中，αt（sj）表示 HMM 在时刻 t 处于状态 j，且 observation = {x1,...,xt}的概率

$$\alpha_t(s_j) = p(\mathbf{x}_1, \ldots, \mathbf{x}_t, S(t) = s_j \mid \boldsymbol{\lambda})$$ ；

$\beta_t(s_j)$: t 时刻处于状态 sj 的话，t 时刻未来 observation 的概率；

且 $p(\mathbf{X} \mid \boldsymbol{\lambda}) = \alpha_T(s_E)$

finally，带入$\gamma_t(s_j)$的定义式有：

$$\gamma_t(s_j) = P(S(t) = s_j \mid \mathbf{X}, \boldsymbol{\lambda}) = \frac{1}{\alpha_T(s_E)}\alpha_t(j)\beta_t(j)$$

http://blog.csdn.net/abcjennifer

好，终于搞定！对应上面的 E-step 目标，只要给定了 observation 和当前 HMM 参数 λ，我们就可以估计$\gamma_t(s_j)$了对吧 (*^__^*)

M-step：根据$\gamma_t(s_j)$重新估计 HMM 参数 λ：

对于 λ 中高斯参数部分，和 GMM 的 M-step 是一样一样的（只不过这里写成向量形式）：

$$\hat{\boldsymbol{\mu}}^j = \frac{\sum_{t=1}^{T}\gamma_t(s_j)x_t}{\sum_{t=1}^{T}\gamma_t(s_j)}$$

$$\hat{\boldsymbol{\Sigma}}^j = \frac{\sum_{t=1}^{T}\gamma_t(s_j)(x_t - \hat{\boldsymbol{\mu}}^j)(x - \hat{\boldsymbol{\mu}}^j)^T}{\sum_{t=1}^{T}\gamma_t(s_j)}$$

对于 λ 中的状态转移概率 aij，定义 C(Si->Sj)为从状态 Si 转到 Sj 的次数，有

$$\hat{a}_{ij} = \frac{C(s_i \to s_j)}{\sum_k C(s_i \to s_k)}$$

实际计算时，定义每一时刻的转移概率$\xi_t(s_i, s_j)$为时刻 t 从 si->sj 的概率：

$$\begin{aligned}\xi_t(s_i, s_j) &= P(S(t) = s_i, S(t+1) = s_j \mid \mathbf{X}, \boldsymbol{\lambda}) \\ &= \frac{P(S(t) = s_i, S(t+1) = s_j, \mathbf{X} \mid \boldsymbol{\lambda})}{p(\mathbf{X}|\Lambda)} \\ &= \frac{\alpha_t(s_i)a_{ij}b_j(\mathbf{x}_{t+1})\beta_{t+1}(s_j)}{\alpha_T(s_E)}\end{aligned}$$

http://blog.csdn.net/abcjennifer

那么就有：

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T}\xi_t(s_i, s_j)}{\sum_{k=1}^{N}\sum_{t=1}^{T}\xi_t(s_i, s_k)}$$

把 HMM 的 EM 迭代过程和要求的参数写专业点，就是这样的：

E step For all time-state pairs

1. Recursively compute the forward probabilities $\alpha_t(s_j)$ and backward probabilities $\beta_t(j)$
2. Compute the state occupation probabilities $\gamma_t(s_j)$ and $\xi_t(s_i, s_j)$

M step Based on the estimated state occupation probabilities re-estimate the HMM parameters: mean vectors $\boldsymbol{\mu}^j$, covariance matrices $\boldsymbol{\Sigma}^j$ and transition probabilities a_{ij}

http://blog.csdn.net/abcjennifer

PS：这个训练 HMM 的算法叫 Forward-Backward algorithm。

一个很好的 reference：点击打开链接

严禁传播 1074