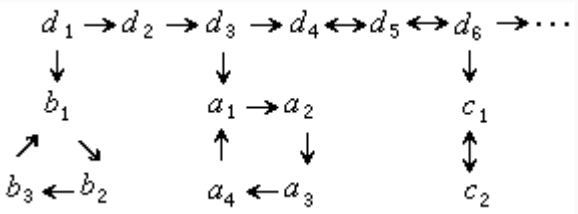
Hidden markov model

1. Markov chain: is a random process that undergoes transitions from one state to another on a state space. It must possess a property that is usually characterized as "memoryless": the probability distribution of the next state depends only on the current state and not on the sequence of events that preceded it. This specific kind of "memorylessness" is called the Markov property. Markov chains have many applications as statistical models of real-world processes.



Markov chain can include many loops.

为随机矩阵，它刻画的马尔可夫链是一个具有反射壁的随机游动。设想一质点的可能位置是直线上的整数点 0，1，…，M，0和M称为壁，它每隔单位时间转移一次，每次向右或左移动一个单位。如果它处在0或M，单位时间后质点必相应地移动到1或M−1，如果它处于0和M之间的 i，则它以概率 p 转移到 $i+1$，以概率 q 转移到 $i-1$。又如果把 P 的第一行换成 $(1, 0, \cdots, 0)$，则此时表示0是吸收壁，质点一旦达到0，它将被吸收而永远处于0。如果不设置壁，质点在直线上的一切整数点上游动，称为自由随机游动，特别当 $f_0 = \frac{c}{2\pi}\sqrt{\frac{p}{mh}}$ 时，称为对称随机游动。

生质的矩阵称为随机矩阵。例如，设 $0<p<1$，$q=1-p$，

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ q & 0 & p & 0 & \cdots & 0 & 0 & 0 \\ 0 & q & 0 & p & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & q & 0 & p \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{pmatrix}$$

memoryless property of Exponential Distribution: 如果灯泡已经用了 t 小时，则他还能用 s 小时的概率和直接从 0 小时开始直接使用 s 小时的概率是一样的

$$P(T > t+s \mid T > t) = \frac{1-(1-e^{-\lambda(t+s)})}{1-(1-e^{-\lambda t})} = e^{-\lambda t} = 1-(1-e^{-\lambda s}) = P(T > s)$$

What is Hidden in HMMs?

Why is this model called a *Hidden* Markov Model? What is hidden from us in this model? The answer is that what is hidden are the states – when we observe the sequence of rolls, we are unable to tell whether or not the casino player is rolling a fair or loaded die, which corresponds to the state of the model; this information about which die the player is rolling affects the outcome, but we are unable to observe it.

The fact that the states are hidden from us is really the crux of HMMs. For example, when we are given a biological sequence, interesting features such as genes and splice sites are hidden from us, and these features are what we wish to find. In the following few lectures, we shall be learning principled methods for finding good sequences of states: although we cannot really catch the casino player with a smoking gun, pinpointing the rolls in which he was cheating, we will be able to determine with high probability whether or not he was using the loaded die; finding good sequences of states is the goal of studying HMMs.

2. Assumptions:

The Markov assumption,

the stationary assumption,

the output independence assumption.

http://jedlik.phy.bme.hu/~gerjanos/HMM/node5.html

Observations are independent with each other, while states are not.

3. Markov problems:

Three Primary Problems/Algorithms for HMMs

1. **Likelihood** - Given an HMM, and an observation sequence O, compute the likelihood $p(O|\lambda)$
2. **Decoding** - Given an observation sequence O and an HMM, $\lambda = (A, B)$, discover the best hidden state sequence Q
3. **Learning (Unsupervised)** - Given an observation sequence O and a set of states for an HMM, learn the HMM parameters A and B.
4. **Learning (Supervised)** - Given an observation sequence O and corresponding hidden state sequence, Q, estimate parameters A and B.

(1)The Evaluation Problem (forward value)

Given an HMM $\boldsymbol{\lambda}$ and a sequence of observations $\boldsymbol{O = o_1, o_2, \ldots, o_T}$, what is the probability that the observations are generated by the model, $\boldsymbol{p\{O|\lambda\}}$?

(2)The Decoding Problem (Viterbi)

Given a model $\boldsymbol{\lambda}$ and a sequence of observations $\boldsymbol{O = o_1, o_2, \ldots, o_T}$, what is the most likely state sequence in the model that produced the observations?

(3)The Learning Problem

- MLE (maximum likelihood estimation)

- Viterbi training(**do not** confuse with Viterbi decoding)
- Baum Welch = forward-backward algorithm

Given a model λ and a sequence of observations $O = o_1, o_2, \ldots, o_T$, how should we adjust the model parameters $\{A, B, \pi\}$ in order to maximize $p\{O|\lambda\}$.

4. Forward algorithm and Viterbi algorithm and backward algorithm

Motivation of forward algorithm:

- Computing $P(O)$ directly is intractable: N^T possible state sequences (N is the number of states and T is the length of the sequence)
- An efficient $O(N^2 T)$ algorithm exists by cleverly caching intermediate values in a *trellis* as we procede from the beginning to the end of the sequence
- We define a set of variables that denote the probability of arriving in state j at time/position t and seeing the observation sequence up to t:

$$\alpha_t(j) = p(o_1, o_2, .., o_t, q_t = j|\lambda)$$

1. Initialization:
$$\alpha_1(j) = a_{0j} b_j(o_1) \ , \ 1 \leq j \leq N$$
2. Recursion:
$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$
3. Termination:
$$P(O|\lambda) = \alpha(q_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF}$$

$$v_t(j) = \max_{q_0,..,q_{t-1}} P(q_0, q_1, .., q_{t-1}, o_1, ..., o_t, q_t = j|\lambda)$$

- Compute the value for each cell recursively:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

1. Initialization:

$$v_1(j) = a_{0j} b_j(o_1), 1 \leq j \leq N$$
$$bt_1(j) = 0$$

2. Recursion:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$
$$bt_t(j) = \arg\max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

3. Termination:

$$v_t(q_F) = \max_{i=1}^{N} v_T(i) * a_{iF} \qquad bt_T(q_F) = \arg\max_{i=1}^{N} v_T(i) * a_{if}$$

Backward values define the probability of seeing the rest of the sequence (one past current position t) given that we are in state i at position t.

$$\beta_t(i) = p(o_{t+1}, ..., o_T | q_t = i, \lambda)$$

1. Initialization: Let

$$\beta_T(i) = a_{iF}, 1 \leq i \leq N$$

2. Recursion:

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

   - For: $1 \leq i \leq N, 1 \leq t \leq T$

3. Termination:

$$p(O|\lambda) = \alpha(q_f) = \beta_0(0) = \sum_{j=1}^{N} a_{0j} b_j(o_1) \beta_1(j)$$

Two equations:

$$p(a|b)p(b|c) = p(a, b|c) \text{ IF } (a \perp c|b)$$
$$p(a, b)p(c|b) = p(a, b, c) \text{ IF } (a \perp c|b)$$

5. Posterior probability vs Viterbi probability

http://stats.stackexchange.com/questions/31119/posterior-probability-vs-viterbi-algorithm

$$\lambda_k(i) = P(q_k = i \mid O, \lambda) \tag{3.18}$$

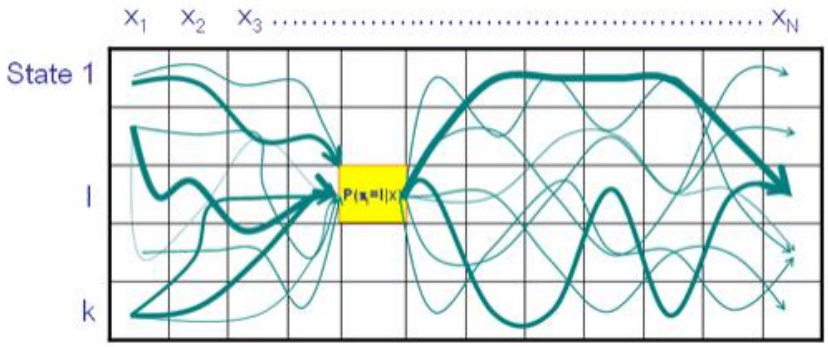
$$\lambda_k(i) = \frac{\alpha_k(i)\beta_k(i)}{P(O \mid \lambda)} \quad k = \overline{1,L}; i = \overline{1,N} \tag{3.19}$$

$$q_k = \arg\max_{1 \le i \le N} \{\lambda_k(i)\} \quad k = \overline{1,L} \tag{3.20}$$

Posterior probability

If aij=0, posterior probability may return some impossible result, since it only care about the posterior probability at each time point, but not the whole sequence.

The following schematic shows what we are doing when computing these posterior probabilities – we are summing the probabilities over all the paths that go through a particular state l.

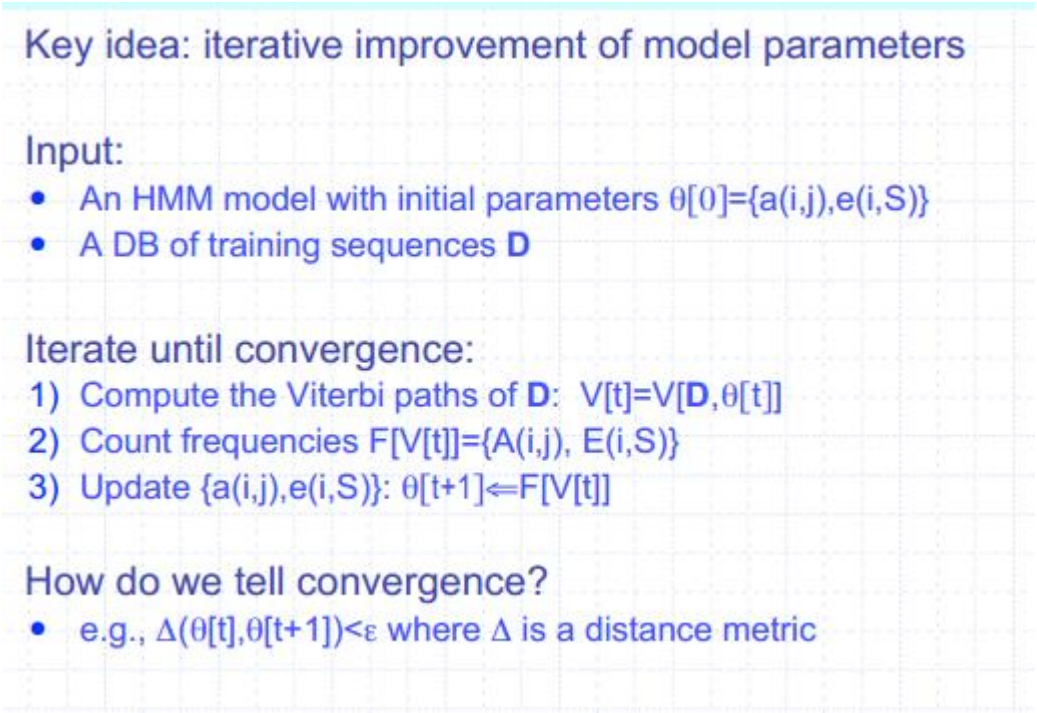


With these posterior probabilities, we can then define the *posterior decoding* parse $\hat{\pi}$ of a sequence $x = x_1 \ldots x_N$ as $\hat{\pi} = \hat{\pi}_1 \ldots \hat{\pi}_N$ where for each i

$$\hat{\pi}_i = \text{argmax}_k \, P(\pi_i = k \mid x)$$

Furthermore, posterior decoding also gives us a confidence for each position in our parse – how likely the underlying state is, and how likely the other states are, in the posterior probabilities we computed. Often this is more informative than a Viterbi parse. However, posterior decoding may give an invalid sequence of states – in the event that some transition probabilities are zero, the parse given by posterior decoding may have probability zero as well. Posterior decoding merely bunches together independent underlying states without regard to whether parses including $\hat{\pi}_i$ and $\hat{\pi}_{i+1}$ for any position i in the parse are distinct and thus it may produce an invalid parse. This is significant, because if for example we are using a HMM to determine whether positions in a DNA sequence were introns or exons, posterior decoding may give us a parse (stating which positions are introns or exons) that could not possibly correspond to a gene as splice sites are missing or because the exons are cut in the middle. We thus have to be careful in interpreting the results when we are using posterior decoding.

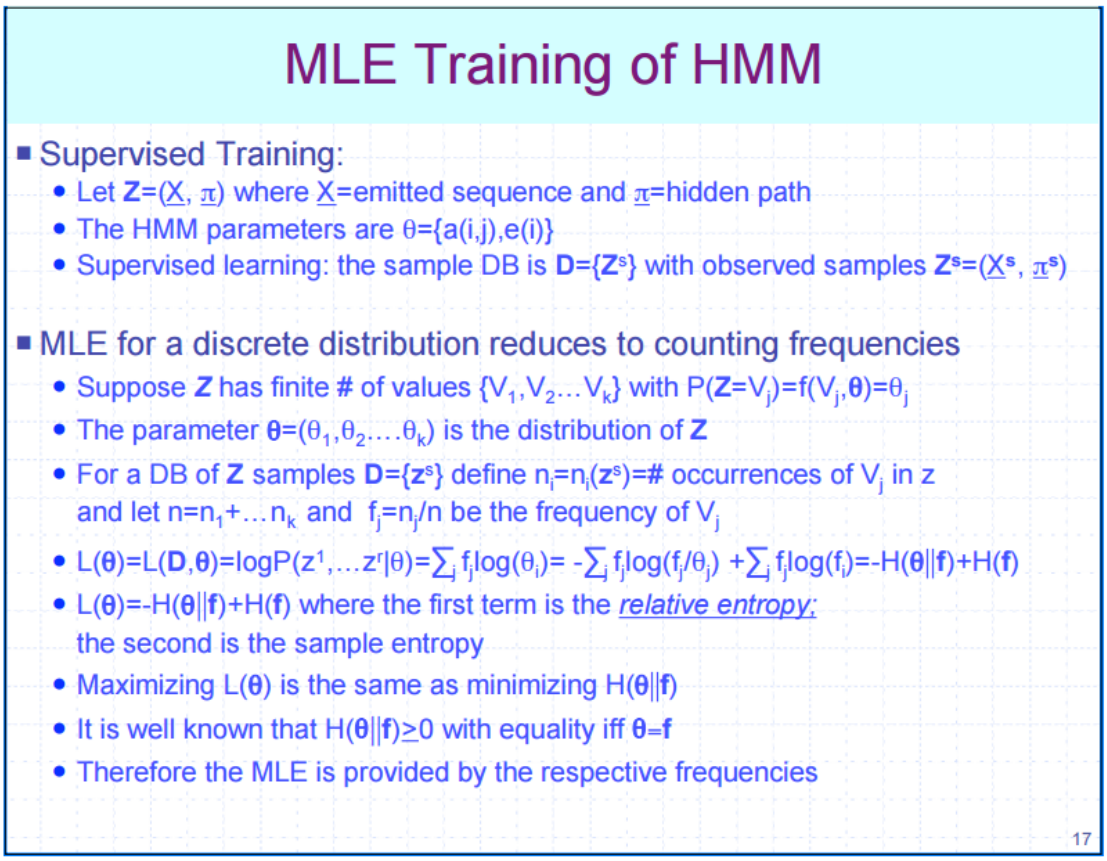
6. Viterbi training vs EM training (both for unsupervised learning)



Viterbi training aims to maximize the probability of the most likely state sequence. Need good initial values. For discrete hmm, EM training is much better, since Viterbi training is influenced by segmentation quality. Viterbi training is cheaper. For some continuous hmm. Viterbi is better.

7. MLE of hmm
http://webcourse.cs.technion.ac.il/236522/Spring2008/ho/WCFiles/class08-m8.pdf

8. Supervised hmm learning vs unsupervised hmm learning
I just wonder if hmm is like naïve bayes. We can prove that counting makes sense in terms of math. Here is the explanation. Usually, we can prove this by using MLE.



For unsupervised hmm, we use expectation instead of counting. We use forward and backward algorithm to calculate expected transition and emission iteratively.

9. Continuous hmm & gmm & how to use ghmm

Input of (multivariable) GMM: weight(i,j), covariance matrix, mean.
Input of continuous hmm: transition matrix, initial matrix.

In each state, continuous hmm has a Gaussian mixture mode, which consists of many Gaussian distributions. In my opinion, it is a little like adding a kernel compared with discrete hmm. In discrete hmm each state has many observations. Here observation is not discrete. We calculate emission probability by calculating y in Gaussian mixture model.

In continuous hmm weight Wk(j,l) that is the probability of the observation to belong to the l-th mixture component in the state j at time k

$$W_k(j,l)=\left[\frac{\alpha_k(j)\beta_k(j)}{P(O|\lambda)}\right]\cdot\left[\frac{v_{jl}f(o_k;\mu_{jl},\Sigma_{jl})}{\sum_{m=1}^{K}v_{jm}f(o_k;\mu_{jm},\Sigma_{jm})}\right] \tag{3.34}$$

$$v_{jl}=\frac{\sum_{k=1}^{L}W_k(j,l)}{\sum_{k=1}^{L}\sum_{m=1}^{K}W_k(j,m)}\quad j=\overline{1,N};l=\overline{1,K} \tag{3.35}$$

$$\mu_{jl}=\frac{\sum_{k=1}^{L}W_k(j,l)\cdot o_k}{\sum_{k=1}^{L}W_k(j,l)} \tag{3.36}$$

$$\Sigma_{jl}=\frac{\sum_{k=1}^{L}W_k(j,l)\cdot(o_k-\mu_{jl})\cdot(o_k-\mu_{jl})^T}{\sum_{k=1}^{L}W_k(j,l)} \tag{3.37}$$

countably infinite number of hidden states.

针对 discrete/continuous hmm:
http://mplab.ucsd.edu/tutorials/hmm.pdf

Continuous
https://onlinecourses.science.psu.edu/stat557/book/export/html/108
http://www.inf.ed.ac.uk/teaching/courses/asr/2012-13/asr03-hmmgmm-4up.pdf