Tutorial on Hidden Markov Models

Javier R. Movellan

1 Notation for discrete HMM

$S = \{S_1, ..., S_N\}$. Set of possible hidden states.

N. Number of distinct hidden states.

$V = \{v_1, ..., v_M\}$. Set of possible external observations.

M. Number of distinct external observations.

$o = (o^1, ..., o^K)$. A sample of sequences of external observations (the training sample). Each element in o is an entire sequence of observations (e.g., a word).

K. Number of sequences (e.g., words) in the training sample.

$o = (o_1, ..., o_T)$. A sequence of external observations (e.g., a word). If there is more than a sequence I use a superscript l.

o_t. A variable representing the external observation at time t. If there is more than a sequence of interest, I use a superscript (e.g., $o_2^4 = 3$ means that in sequence number 4, at time 2 we observe v_3).

T. The number of time steps in the sequence .

$q = (q^1, ..., q^K)$. Collection of sequences of internal states (internal state sequences that may correspond to each sequence in the training sequence).

$q = (q_1, ..., q_T)$. A sequence of internal states. If there is more than a sequence of interest I use a superscript to denote the sequence of interest.

q_t. A variable representing the internal state at time t. If there is more than a sequence I use a superscript (e.g., $q_2^4 = 3$ means that the system is in state S_3 at time 2 in sequence number 4 of the training sample).

$\lambda = (A, B, \pi)$. A hidden Markov model as defined by its A, B and π matrices.

$a_{ij} = P_\lambda(q_{t+1} = j | q_t = i)$. State transition probability for model λ.

$A = \{a_{ij}\}$. The NxN matrix of transition probabilities.

$b_j(k) = P_\lambda(o_t = k | q_t = j)$. Emission probability for observation v_k by state S_j.

$B = \{b_j(k)\}$. The MxN matrix of state to observations probabilities.

$\pi_i = P_\lambda(q_1 = i)$. Initial state probability for model λ.

$\pi = \{\pi_i\}$. The vector of initial state probabilities.

$Q(\lambda, \bar{\lambda}) = \sum_q P_\lambda(q|0) log\ P_{\bar{\lambda}}(qo)$. The auxiliary function maximized by the E-M algorithm. λ represents the current model, $\bar{\lambda}$ represents the new model under consideration.

$Q^l(\lambda, \bar{\lambda})$ The E-M function restricted to the sequence o^l from the training sample.

$\alpha_t(i) = P_\lambda(o_1 ... o_t\ q_t = i)$. The forward variable for the sequence o at time t for state i. If there is more than one sequence of interest I use a superscript to denote the sequence.

$\beta_t(i) = P_\lambda(o_{t+1} ... o_T | q_t = i)$. The scaled backward variable for the sequence o at time t for state i. If there is more than one sequence of interest I use a superscript to denote the sequence.

$\hat{\alpha}_t(i)$. The scaled forward variable for the sequence o at time t for state i. If there is more than one sequence of interest I use a superscript to denote the sequence.

$\hat{\beta}_t(i)$. The scaled backward variable for the sequence o at time t for state i. If there is more

than one sequence of interest I use a superscript to denote the sequence.

$c_1 ... c_T$. The scaling coefficients in the scaled forward and backward algorithm for the sequence o^l. If there is more than one sequence of interest I use a superscript to denote the sequence.

$\gamma_t(i) = P_\lambda(q_t = i|o)$ If there is more than one sequence of interest I use a superscript to denote the sequence.

$\xi_t(i,j) = P_\lambda(q_t = i \; q_{t+1} = j|o)$. If there is more than one sequence of interest I use a superscript to denote the sequence.

2 EM training with Discrete Observation Models

In this section we review two methods for training standard HMM models with discrete observations: E-M training and Viterbi training.

2.1 The E-M auxiliary function

Let λ represent the current model and $\bar{\lambda}$ represent a candidate model. Our objective is to make $P_{\bar{\lambda}}(o) \geq P_\lambda(o)$, or equivalently $log\ P_{\bar{\lambda}}(o) \geq log\ P_\lambda(o)$.

Due to the presence of stochastic constraints (e.g., $a_{ij} \geq$ and $\sum_j a_{ij} = 1$) it turns out to be easier to maximize an auxiliary function $Q(\cdot)$ rather than to directly maximize $log\ P_{\bar{\lambda}}$. The E-M auxiliary function is defined as follows:

$$Q(\lambda, \bar{\lambda}) = \sum_q P_\lambda(q|o) log\ P_{\bar{\lambda}}(qo) \tag{1}$$

Here we show that $Q(\lambda, \bar{\lambda}) \geq Q(\lambda, \lambda) \rightarrow log\ P_{\bar{\lambda}}(o) \geq log\ P_\lambda(o)$.

For any model λ or $\bar{\lambda}$ it must be true that

$$P_{\bar{\lambda}}(o) = \frac{P_{\bar{\lambda}}(oq)}{P_{\bar{\lambda}}(q|o)} \tag{2}$$

or $logP_{\bar{\lambda}}(o) = logP_{\bar{\lambda}}(oq) - logP_{\bar{\lambda}}(q|o)$.

Also,

$$logP_{\bar{\lambda}}(o) = \sum_q P_\lambda(q|o) logP_{\bar{\lambda}}(o) \tag{3}$$

since $logP_{\bar{\lambda}}(o)$ is a constant. Thus, from equation 2 it follows that

$$log\ P_{\bar{\lambda}}(o) = \sum_q P_\lambda(q|o) logP_{\bar{\lambda}}(oq) - \sum_q P_\lambda(q|o) logP_{\bar{\lambda}}(q|o) \tag{4}$$

$$= Q(\lambda, \bar{\lambda}) - \sum_q P_\lambda(q|o) logP_{\bar{\lambda}}(q|o) \tag{5}$$

Applying the $Q(\cdot, \cdot)$ function to $(\lambda, \bar{\lambda})$ and to (λ, λ), it follows that,

$$Q(\lambda, \bar{\lambda}) - Q(\lambda, \lambda) = log\ P_{\bar{\lambda}}(o) - log\ P_\lambda(o) - KL(\lambda, \bar{\lambda}) \tag{6}$$

where $KL(\cdot, \cdot)$ is the Kullback-Leibler criterion (relative entropy) of the probability distribution $P_\lambda(q|o)$ with respect to the probability distribution $P_{\bar{\lambda}}(q|o)$

$$KL(\lambda, \bar{\lambda}) = \sum_q P_\lambda(q|o) log \frac{P_\lambda(q|o)}{P_{\bar{\lambda}}(q|o)} \tag{7}$$

Rearranging terms,

$$log\ P_{\bar{\lambda}}(o) - log\ P_\lambda(o) = Q(\lambda, \bar{\lambda}) - Q(\lambda, \lambda) + KL(\lambda, \bar{\lambda}) \tag{8}$$

and since the KL criterion is always positive, it follows that if

$$Q(\lambda, \bar{\lambda}) - Q(\lambda, \lambda) \geq 0 \tag{9}$$

then

$$log\ P_{\bar{\lambda}}(o) log\ P_\lambda(o) \geq 0 \tag{10}$$

2.2 The overall training sample E-M function

We defined the overall E-M function

$$Q(\lambda, \bar{\lambda}) = \sum_q P_\lambda(q|o) log\ P_{\bar{\lambda}}(qo) \tag{11}$$

with o including the entire set of sequences in the training sample. Assuming that the sequences are independent, it follows that

$$Q(\lambda, \bar{\lambda}) = \sum_q P_\lambda(q|o) \sum_l log\ P_{\bar{\lambda}}(q^l o^l) = \sum_l \sum_q (log\ P_{\bar{\lambda}}(q^l o^l)) P_\lambda(q|o) \tag{12}$$

And since each state sequence q^l depends only on the corresponding observation sequence o^l it follows that

$$Q(\lambda, \bar{\lambda}) = \sum_l \sum_{q^l} \sum_{q-q^l} (log\ P_{\bar{\lambda}}(q^l o^l)) \prod_{m=1}^{K} P_\lambda(q^m|o) \tag{13}$$

where $q - q^l = (q^1, ..., q^{l-1}, q^{l+1}, ..., q^K)$ represents an entire collection of sequences except for the sequence q^l. Thus

$$Q(\lambda, \bar{\lambda}) = \sum_l \sum_{q^l} \sum_{q-q^l} (log\ P_{\bar{\lambda}}(q^l o^l)) P_\lambda(q^l|o^l) \prod_{m \neq l}^{K} P_\lambda(q^m|o) = \tag{14}$$

$$= \sum_l \sum_{q^l} (log\ P_{\bar{\lambda}}(q^l o^l)) P_\lambda(q^l|o^l) \sum_{q-q^l} \prod_{m \neq l} P_\lambda(q^m|o) \tag{15}$$

and since

$$\sum_{q-q^l} \prod_{m \neq l}^{K} P_\lambda(q^m|o) = 1 \tag{16}$$

it follows that the E-M function can be decomposed into additive E-M functions, one per observation sequence in the training sample:

$$Q(\lambda, \bar{\lambda}) = \sum_l \sum_{q^l} P_\lambda(q^l|o^l)\ log\ P_{\bar{\lambda}}(q^l o^l) = \sum_l Q^l(\lambda, \bar{\lambda}) \tag{17}$$

2.3 Maximizing the E-M function

For simplicity let us start with the case in which there is a single sequence. The results easily generalize to multiple sequences. Since we work with a single sequence we may drop the l superscript.

$$Q(\lambda, \bar{\lambda}) = \sum_q P_\lambda(q|o) log\ P_{\bar{\lambda}}(qo) \tag{18}$$

And since,

$$P_{\bar{\lambda}}(qo) = \bar{\pi}_{q_1} \bar{b}_{q_1}(o_1) \bar{a}_{q_1 q_2} \bar{b}_{q_2}(o_2) \bar{a}_{q_2 q_3} ... \tag{19}$$

it follows that,

$$Q(\lambda, \bar{\lambda}) = \sum_q P_\lambda(q|o) log\ \bar{\pi}_{q_1} + \tag{20}$$

$$+ \sum_{t=1}^{T_l} \sum_q P_\lambda(q|o) log\ \bar{b}_{q_t}(o_t) + \tag{21}$$

$$+ \sum_{t=1}^{T_l - 1} \sum_q P_\lambda(q|o) log\ \bar{a}_{q_t q_{t+1}} \tag{22}$$

The first term can be expressed as follows

$$\sum_q P_\lambda(q|o) log\ \bar{\pi}_{q_1} = \sum_j log\ \bar{\pi}_j \sum_q P_\lambda(q|o) \delta(j, q_1) \tag{23}$$

where $\delta(j, q_1)$ tells us to include only those cases in which $q_1 = j$. Therefore,

$$\sum_q P_\lambda(q|o) \delta(j, q_1) = P_\lambda(q_1 = j|o) = \gamma_1(j) \tag{24}$$

The second term can be expressed as follows

$$\sum_{t=1}^{T_l} \sum_q P_\lambda(q|o) log\ \bar{b}_{q_t}(o_t) = \sum_{t=1}^{T_l} \sum_i \sum_j log\ \bar{b}_i(j) \sum_q P_\lambda(q|o) \delta(i, q_t) \delta(j, o_t) \tag{25}$$

where $\delta(i, q_t)\delta(j, o_t)$ tells us to include only those cases for which $q_t = i$ and $o_t = j$. Therefore,

$$\sum_q P_\lambda(q|o) \delta(i, q_t) \delta(j, o_t) = P_\lambda(q_t = i|o) \delta(o_t, j) = \gamma_t(i) \delta(o_t, j) = \tag{26}$$

The third term can be expressed as follows

$$\sum_{t=1}^{T_l - 1} \sum_q P_\lambda(q|o) log\ \bar{a}_{q_t q_{t+1}} = \sum_{t=1}^{T_l - 1} \sum_i \sum_j log\ \bar{a}_{ij} \sum_q P_\lambda(q|o) \delta(i, q_t) \delta(j, q_{t+1}) \tag{27}$$

where $\delta(i, q_t)\delta(j, q_{t+1})$ tells us to include only those cases for which $q_t = i$ and $q_{t+1} = j$. Therefore,

$$\sum_q P_\lambda(q|o)\delta(i,q_t)\delta(j,q_{t+1}) = P_\lambda(q_t = i\ q_{t+1} = j|o) = \xi_t(i,j) \tag{28}$$

Putting it together

$$Q(\lambda,\bar{\lambda}) = \sum_j \gamma_1(j)\ log\ \bar{\pi}_j + \tag{29}$$

$$+\sum_i \sum_j log\ \bar{b}_i(j)\ (\sum_{t=1}^{T_l} \gamma_t(i)\delta(o_t,j)) + \tag{30}$$

$$+\sum_i \sum_j\ log\ \bar{a}_{ij}\ (\sum_{t=1}^{T_l-1} \xi_t(i,j)) \tag{31}$$

When there is more than a sequence, we just need to add up over sequences to obtain the overall $Q(\cdot,\cdot)$ function

$$Q(\lambda,\bar{\lambda}) = \sum_j log\ \bar{\pi}_j(\sum_l \gamma_1^l(j)) + \tag{32}$$

$$+\sum_i \sum_j log\ \bar{b}_i(j)\ (\sum_l \sum_{t=1}^{T_l} \gamma_t^l(i)\delta(o_t^l,j)) + \tag{33}$$

$$+\sum_i \sum_j\ log\ \bar{a}_{ij}(\sum_l \sum_{t=1}^{T_l-1} \xi_t^l(i,j)) \tag{34}$$

Note that the part of the overall $Q(\lambda,\bar{\lambda})$ function dependent on $\bar{\pi}_j$ is of the form $w_j\ log\ x_j$ with $x_j = \bar{\pi}_j$ and

$$w_j = \sum_{l=1}^{K} \gamma_1^l(j) \tag{35}$$

with constraints $\sum_j x_j = 1$, and $x_j \geq 0$.

Equivalently, the part of the overall $Q(\lambda,\bar{\lambda})$ function dependent of $\bar{b}_i(j)$ is of the form $w_j log\ x_j$ with $x_j = \bar{b}_i(j)$ and

$$w_j = \sum_{l=1}^{K} \sum_{t=1}^{T_l} \gamma_t^l(j)\delta(o_t^l,j) \tag{36}$$

with constraints $\sum_j x_j = 1$, and $x_j \geq 0$.

Finally, the part of the overall $Q(\lambda,\bar{\lambda})$ function dependent of $\bar{a}_{ij}$ is also of the form $w_j log\ x_j$ with $x_j = \bar{a}_{ij}$ and

$$w_j = \sum_{l=1}^{K} \sum_{t=1}^{T_l-1} \xi_t^l(i,j) \tag{37}$$

with constraints $\sum_j x_j = 1$, and $x_j \geq 0$.

It is easy to show that the maximum of a function of the form $w_j log\ x_j$ with constraints that $x_j \geq 0$ and $\sum_j x_j = 1$ is achieved for

$$x_j = \frac{w_j}{\sum_j w_j} \tag{38}$$

The parameters of the new model $\bar{\lambda}$ that maximize the overall $Q(\lambda, \bar{\lambda})$ function easily follow:

$$\boxed{\begin{array}{lcl} \bar{\pi}_i & = & \frac{\sum_{l=1}^{K} \gamma_1(i)}{K} \\ \bar{b}_i(j) & = & \frac{\sum_{l=1}^{K} \sum_{t=1}^{T_l} \gamma_t(i) \delta(o_t^l, j)}{\sum_{l=1}^{K} \sum_{t=1}^{T_l} \gamma_t^l(i)} \\ \bar{a}_{ij} & = & \frac{\sum_{l=1}^{K} \sum_{t=1}^{T_l - 1} \xi_t(i,j)}{\sum_{l=1}^{K} \sum_{t=1}^{T_l - 1} \gamma_t^l(i)} \end{array}} \tag{39}$$

2.4 Obtaining the E-M parameters from the scaled forward and backward algorithms

Section under construction

2.5 MM Training

MM training (Maximization Maximization) may be seen as an approximation to EM training. In MM training we basically substitute the Expected value operation by a Max operation thus the MM name. Other names for these algorithms are: Viterbi training (because we use the Viterbi algorithm to do the Max operation) and segmented K-means (K-means is a classical clustering method that belongs to the MM family).

In Viterbi-based decoding, the degree of match between a model λ and an observation sequence o^l is defined as $\rho(\lambda, o^l) = \max_{q^l} log\ P_\lambda(q^l\ o^l) = log\ P_\lambda(\hat{q}^l\ o^l)$. We have seen how for a fixed model λ, the Viterbi recurrence can be used to find $\hat{q}^l$ and $\rho(\lambda, o^l)$. When there is more than one training sequences they are assumed independent and $\rho(\lambda, o) = \sum_{l=1}^{K} \rho(\lambda, o^l)$. Thus, the optimal states can be found by applying Viterbi decoding independently for each of the training sequences.

In MM training the objective is to find an optimal point of $log\ P_\lambda(q^l\ o^l)$ with the model λ and the state sequence q as optimizing variables.

To begin with, assume that Viterbi decoding has found the best sequence of hidden states for the current λ model: $\hat{q} = (\hat{q}^1, ..., \hat{q}^K)$. Once we have $\hat{q}$ we find a new model $\bar{\lambda}$ such that $log\ P_{\bar{\lambda}}(o\ \hat{q}) \geq log\ P_\lambda(o\ \hat{q})$. To do so simply define a dummy model $\hat{\lambda}$ such that $P_{\hat{\lambda}}(qo) = \delta(\hat{q}, o)$, thus the dummy model is such that only state sequence $\hat{q}$ can co-occur with observation sequence o.

For such model, $P_{\hat{\lambda}}(q_1 = j|o^l) = \delta(i, \hat{q}^l)$, $P_{\hat{\lambda}}(q_t = i|o^l) = \delta(i, \hat{q}_t)$, and $P_{\hat{\lambda}}(q_t = i\ q_{t+1} = j|o^l) = \delta(i, \hat{q}_t)\delta(j, \hat{q}_{t+1}^l)$. Also note that $Q(\hat{\lambda}, \bar{\lambda}) = log\ P_{\bar{\lambda}}(o\ \hat{q})$, the function we want

to optimize. Thus, substituting the $\hat{\lambda}$ parameters in the standard E-M formulas guarantees that $Q(\hat{\lambda}, \bar{\lambda}) \geq Q(\hat{\lambda}, \lambda)$ or $log\ P_{\bar{\lambda}}(o\ \hat{q}) \geq log\ P_{\lambda}(o\ \hat{q})$

Thus, the MM training rules are as follows:

$$\boxed{\begin{array}{lcl} \bar{\pi}_i & = & \frac{\sum_{l=1}^{K} \delta(i,\hat{q}_1^l)}{K} \\ \\ \bar{b}_i(j) & = & \frac{\sum_{l=1}^{K} \sum_{t=1}^{T_l} \delta(i,\hat{q}_t^l)\delta(j,o_t^l)}{\sum_{l=1}^{K} \sum_{t=1}^{T_l} \delta(i,\hat{q}_t^l)} \\ \\ \bar{a}_{ij} & = & \frac{\sum_{l=1}^{K} \sum_{t=1}^{T_l-1} \delta(i,\hat{q}_t^l)\delta(j,\hat{q}_{t+1}^l)}{\sum_{l=1}^{K} \sum_{t=1}^{T_l-1} \delta(i,\hat{q}_t^l)} \end{array}} \tag{40}$$

Note that Viterbi decoding maximizes $log\ P_{\lambda}(q, o)$ with respect to q for a fixed model (the first M step) then we maximizes $log\ P_{\lambda}(q, o)$ with respect to λ, for a fixed q (the second M step). Since we are always maximizing with respect to some variables, $log\ P_{\lambda}(q\ o)$ can only increase and convergence to a local maximum is guaranteed.

3 Notation for Continuous Density Models

The notation for the continuous case is the same as the discrete case with the following additional terms.

P Number of dimensions per observation (e.g. cepstral coefficients): $o_t = (o_{t_1}...ot_P)$.

M. Number of clusters within a state.

$V = \{v_{11}, ..., v_{1M}..., v_{N1}..., v_{NM}\}$. Set of possible clusters of external observations (M clusters per state). Each cluster v_{ij} is identified by a state index i and a cluster index j.

m_t variable identifying the cluster index of the cluster that occurred at time t. For example if cluster v_{ik} occurred at time t then $q_t = i$ and $m_t = k$.

$m = (m_1...m_t)$. A sequence of cluster indexes, one per time step.

$g_{ik} = P_\lambda(m_t = k|q_t = j)$. Emission probability (gain) of cluster kv_{ik} by state S_j.

$\phi(\cdot)$. A kernel function (e.g. Gaussian) to model the probability density of a cluster of observations produced by a state.

$b_j(o_t) = P_\lambda(o_t|q_t = j)$.

$\mu_{ik} = (\mu_{ik1}, ..., \mu_{ikP})$. The centroid or prototype of cluster v_{ik}.

$\Sigma_{ik} = \begin{pmatrix} \sigma^2_{ik1} & \sigma_{ik12} & ... & \sigma_{ik1p} \\ \sigma_{ik21} & \sigma^2_{ik2} & ... & \sigma_{ik2p} \\ ... & & & \\ \sigma_{ikp1} & \sigma_{ikp2} & ... & \sigma^2_{ikp} \end{pmatrix}$ The covariance matrix of cluster v_{ik}.

σ^2_{ikn}. The variance of the l^{th} dimension (e.g. cepstral) within cluster v_{ik}, a diagonal element of Σ_{ik}.

σ_{iknm}, The covariance between dimension l and dimension m within the cluster v_{ik} an off-diagonal elements of Σ_{ik}. Usually assumed zero.

4 Continuous Observation Models

In this section we study the E-M and Viterbi training procedures for continuous observation HMMs.

4.1 Mixtures of Densities

In the discrete case the observations are discrete, represented by an integer. In the continuous case the observations at each time step are P-dimensional real-valued vectors (e.g. cepstrals coefficients). In our notation $o_t = (o_{t1}, ..., o_{tP})$.

The continuous observation model produces sequences of observations in the following way: At each time step the system generates a hidden state q_t according to a a state to state transition probability distribution $a_{q_{t-1}q_t}$. Once q_t has been generated, the system generates a hidden cluster m_t according to a state to cluster emission probability distribution $g_{q_t m_t}$. Once the hidden cluster has been determined, an observation vector is produced probabilistically according to some kernel probability distribution (e.g. Multivariate Gaussian). We can think of the clusters as low level hidden states embedded within high level hidden states q_t. For example, the high level hidden states may represent phonemes and the low-level hidden clusters may represent acoustic categories within the same phoneme. For simplicity each state is assumed to have the same number of clusters (M) but the set of clusters is different from state to state. Thus, there is a total of NxM clusters, M per state.

We represent cluster k of state S_j as v_{jk} and we use the variable m_t to identify the cluster number within a state at time t. Thus if $q_t = i$ and $m_t = k$ it means that at time t cluster v_{jk} occurred.

If we know the cluster at time t, the probability density of a vector of continuous observations (e.g. cepstral coefficients) is modeled by a kernel function, usually a multivariate Gaussian

$$P_\lambda(o_t|v_{jk}) = P_\lambda(o_t|q_t = j\ m_t = k) = \phi(o_t, \mu_{jk}, \Sigma_{jk}) \tag{41}$$

Where $\phi(\cdot)$ is the kernel function (e.g. multivariate Gaussian) $\mu_{jk} = (\mu_{jk1}, ..., \mu_{jkP})$ is a centroid or prototype that determines the position of the cluster in P-dimensional space, and

$$\Sigma_{jk} = \begin{pmatrix} \sigma^2_{jk1} & \sigma_{jk12} & ... & \sigma_{jk1P} \\ \sigma_{jk21} & \sigma^2_{jk2} & ... & \sigma_{jk2P} \\ ... & & & \\ \sigma_{jkP1} & \sigma_{jkP2} & ... & \sigma^2_{jkP} \end{pmatrix} \tag{42}$$

is a covariance matrix that determines the width and tilt of the cluster in P-dimensional space. The diagonal terms $\{\sigma^2_{jk1}...\sigma^2_{jkP}\}$ are the cluster variances for each dimension. They determine the spread of the cluster on each dimension. The off-diagonal elements are known as the cluster covariances and they determine the tilt of the cluster. For the Gaussian case, the kernel function is

$$\phi(o_t, \mu_{jk}, \Sigma_{jk}) = \frac{1}{(2\pi)^{P/2}|\Sigma_{jk}|^{1/2}} e^{-d(o_t, \mu_{jk})} \tag{43}$$

where $|\Sigma_{jk}|$ is the determinant of the variance matrix, and $d(o_t, \mu_{jk})$ is known as the Mahalanobis distance between the observation and the kernel's centroid.

$$d(o_t, \mu_{jk}) = \frac{1}{2}(o_t - \mu_{jk})\Sigma^{-1}_{jk}(o_t - \mu_{jk})' \tag{44}$$

Since the covariance matrices Σ_{jk} are symmetric, each kernel is defined by $\frac{P(P+3)}{2}$ parameters (P for the centroids and $P(P+1)/2$ for the variances). In practice the off-diagonal variances are assumed zero, reducing the number of parameters per kernel to 2P. In such case the determinant $|\Sigma_{jk}|$ is just the product of P scalar variances $|\Sigma_{jk}| = \prod_{l=1}^{P} \sigma^2_{jkl}$, and the Mahalanobis distance becomes a scaled Euclidean distance:

$$d(o_t, \mu_{jk}) = \frac{1}{2}\sum_{l=1}^{P} \frac{(o_{tl} - \mu_{jkl})^2}{\sigma^2_{jkl}} \tag{45}$$

Given a state S_j, the system randomly chooses one of its M possible clusters with state to cluster emission probability $P(m_t = k|q_t = j)$. This probability is assumed independent of t and thus it can be represented by a parameter with no time index. In our notation $P(m_t = k|q_t = j) = g_{jk}$, the gain of the k^{th} cluster embedded in state S_j.

Thus, the overall probability density of the observations generated by a state S_j is given by a weighted mixture of kernel functions.

$$P_\lambda(o_t|q_t = j) = \sum_{l=1}^{M} P(m_t = l|q_t = j)P(o_t|q_t = j,\ m_t = k) = \tag{46}$$

or

$$b_j(o_t) = \sum_{k=1}^{M} g_{jk}\phi(o_t, \mu_{jk}, \Sigma_{jk}) \tag{47}$$

4.2 Forward and backward variables

The un-scaled and the scaled algorithms work the same as in the discrete case. Only now the emission probability terms $b_j(o_t)$ are modeled by a mixture of densities.

$$b_j(o_t) = \sum_{k=1}^{M} g_{jk}\phi(o_t, \mu_{jk}, \Sigma_{jk}) \tag{48}$$

4.3 EM Training

In the continuous case the clusters are low level hidden states m_t embedded within high level hidden states q_t. Thus, the E-M function is defined over all possible q m sequences of high-level and low-level hidden states

$$Q(\lambda, \bar{\lambda}) = \sum_q \sum_m P_\lambda(qm|o) log\ P_{\bar{\lambda}}(qmo) \tag{49}$$

and since

$$P_{\bar{\lambda}}(qmo) = \bar{\pi}_{q_1}\bar{g}_{q_1 m_1}\phi(o_1, \bar{\mu}_{q_1 m_1}, \bar{\Sigma}_{q_1 m_1}), ..., \bar{a}_{q_{T-1}q_T}\bar{g}_{q_T m_T}\phi(o_T, \bar{\mu}_{q_T m_T}, \bar{\Sigma}_{q_T m_T}) \tag{50}$$

it follows that

$$Q(\lambda, \bar{\lambda}) = \sum_q \sum_m P_\lambda(qm|o) log\ \bar{\pi}_{q_1} + \tag{51}$$

$$\sum_{t=1}^{T-1} \sum_q \sum_m P_\lambda(qm|o) log\ \bar{a}_{q_t q_{t+1}} + \tag{52}$$

$$+ \sum_{t=1}^{T} \sum_q \sum_m P_\lambda(qm|o) log\ \bar{g}_{q_t m_t} \tag{53}$$

$$+ \sum_{t=1}^{T} \sum_q \sum_m P_\lambda(qm|o) log\ \phi(o_t, \bar{\mu}_{q_t m_t}, \bar{\Sigma}_{q_t m_t}) \tag{54}$$

Since the factors in the first two terms are independent of m they simplify into

$$\sum_q \sum_m P_\lambda(qm|o) log\ \bar{\pi}_{q_1} = \sum_q P_\lambda(q|o) log\ \bar{\pi}_{q_1} \tag{55}$$

and

$$\sum_{t=1}^{T-1} \sum_q \sum_m P_\lambda(qm|o) log\ \bar{a}_{q_t q_{t+1}} = \sum_{t=1}^{T-1} \sum_q P_\lambda(q|o) log\ \bar{a}_{q_t q_{t+1}} \tag{56}$$

These terms are identical as in the discrete case and thus the same training rules for initial state probabilities and for state transition probabilities apply here.

To find the training formulas for the cluster gains, we focus on the part of $Q(\cdot)$ dependent on the gain terms. This part can be transformed as follows

$$\sum_{t=1}^{T} \sum_q \sum_m P_\lambda(qm|o) log\ \bar{g}_{q_t}(m_t) = \tag{57}$$

$$= \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{q} \sum_{m} P_\lambda(qm|o) log\ \bar{g}_{ik} \delta(i, q_t) \delta(j, m_t) \tag{58}$$

$$= \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{k=1}^{M} P_\lambda(q_t = i\ m_t = k|o) log\ \bar{g}_{ik} \tag{59}$$

Thus the part of $Q(\cdot)$ that depends on $\bar{g}_{ik}$ is of the form $w_j\ log\ x_j$ with $x_j = \bar{g}_{ik}$ and

$$w_j = \sum_{t=1}^{T} P_\lambda(q_t = i\ m_t = k|o) \tag{60}$$

with constraints $\sum_j x_j = 1$ and $x_j \leq 0$, with maximum achieved for

$$x_j = \frac{w_j}{\sum_j w_j} \tag{61}$$

Thus

$$\bar{g}_{ik} = \frac{\sum_{t=1}^{T} P_\lambda(q_t = i\ m_t = k|o)}{\sum_{t=1}^{T} \sum_{k=1}^{M} P_\lambda(q_t = i\ m_t = k|o)} = \frac{\sum_{t=1}^{T} P_\lambda(q_t = i\ m_t = k|o)}{\sum_{t=1}^{T} P_\lambda(q_t = i|o)} \tag{62}$$

To find the learning rules for the centroids and variances we focus on the part of $Q(\cdot)$ that depends on the cluster centroids and variances, which is given by the following expression:

$$\sum_{t=1}^{T} \sum_{q} \sum_{m} P_\lambda(qm|o) log\ \phi(o_t, \bar{\mu}_{q_t m_t}, \bar{\Sigma}_{q_t m_t}) \tag{63}$$

This expression can be transformed as follows

$$\sum_{t=1}^{T} \sum_{q} \sum_{m} P_\lambda(qm|o) log\ \phi(o_t, \bar{\mu}_{q_t m_t}, \bar{\Sigma}_{q_t m_t}) = \tag{64}$$

$$= \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{k=1}^{M} \sum_{q} \sum_{m} P_\lambda(qm|o) log\ \phi(o_t, \bar{\mu}_{ik}, \bar{\Sigma}_{ik}) \delta(i, q_t) \delta(k, m_t) \tag{65}$$

$$= \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{k=1}^{M} P_\lambda(q_t = i\ m_t = k|o) log\ \phi(o_t, \bar{\mu}_{ik}, \bar{\Sigma}_{ik}) \tag{66}$$

Thus, at a maximum,

$$\frac{\partial}{\partial \bar{\mu}_{ikn}} \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{k=1}^{M} P_\lambda(q_t = i\ m_t = k|o) log\ \phi(o_t, \bar{\mu}_{ik}, \bar{\Sigma}_{ik}) = 0 \tag{67}$$

and since

$$\frac{\partial log\ \phi(o_t, \bar{\mu}_{ik}, \bar{\Sigma}_{ik})}{\partial \bar{\mu}_{ikn}}) = \frac{1}{\bar{\sigma}^2_{ikl}} (o_{tl} - \bar{\mu}_{ikn}) \tag{68}$$

it follows that at a maximum

$$\sum_{t=1}^{T} P_\lambda(q_t = i\ m_t = k|o)(o_{tl} - \bar{\mu}_{ikn}) = 0 \tag{69}$$

or

$$\bar{\mu}_{ikn} = \frac{\sum_{t=1}^{T} P_\lambda(q_t = i\ m_t = k|o)\ o_{tn}}{\sum_{t=1}^{T} P_\lambda(q_t = i|o)} \tag{70}$$

A similar argument can be made for the diagonal variance σ^2_{ikl}. In this case

$$\frac{\partial log\ \phi(o_t, \bar{\mu}_{ik}, \bar{\Sigma}_{ik})}{\partial \bar{\sigma}^2_{ikn}} = -\frac{1}{2\bar{\sigma}^2_{ikn}}(1 - \frac{(o_{tn} - \bar{\mu}_{ikn})^2}{\bar{\sigma}^2_{ikn}}) \tag{71}$$

Thus, at a maximum

$$\sum_{t=1}^{T} P_\lambda(q_t = i\ m_t = k|o)(1 - \frac{(o_{tl} - \bar{\mu}_{ikn})^2}{\bar{\sigma}^2_{ikn}}) = 0 \tag{72}$$

from which the re-estimation formula easily follows:

$$\bar{\sigma^2}_{ikn} = \frac{\sum_{t=1}^{T} P_\lambda(q_t = i\ m_t = k|o)(o_{tl} - \bar{\mu}_{ikn})^2}{\sum_{t=1}^{T} P_\lambda(q_t = i|o)} \tag{73}$$

Training for the mixture gains, mixture centroids, and mixture variances requires the $P(q_t = jm_t = k|o)$ terms, for $t = 1...T\ , j = 1..N\ , k = 1...M$. To obtain these terms note the following:

$$P(q_t = j\ m_t = k|o) = P(q_t = j|o)P(m_t = k|q_t = j\ o) = \tag{74}$$

$$= P(q_t = j|o)P(m_t = k|q_t = j\ o_t) \tag{75}$$

$$= P(q_t = j|o)\frac{P(o_t m_t = k|q_t = j)}{P(o_t|q_t = j)} \tag{76}$$

$$= P(q_t = j|o)\frac{P(m_t = k|q_t = j)P(o_t|q_t = jm_t = k)}{P(o_t|q_t = j)} \tag{77}$$

Thus

$$P(q_t = jm_t = k|o) = P(q_t = j|o)\frac{g_{jk}\ \phi(o_t, \mu_{ik}, \Sigma_{ik})}{b_j(o_t)} \tag{78}$$

As in the discrete case, the $P(q_t = j|o)$ can be obtained through the scaled feed-forward algorithm.

For the case with multiple training sequences the overall $Q(\cdot)$ decomposes into additive $Q^l(\cdot)$, one per training sequence. As a consequence we have to add in the numerator and denominator of the training formulas the effects of each training sequence.

Summarizing, the E-M learning rules for the mixture of Gaussian densities case with diagonal covariance matrices are as follows:

$$
\begin{aligned}
\bar{\pi}_i &= \frac{\sum_{l=1}^{K} P_\lambda(q_1{=}j|o^l)}{K} \\
\bar{a}_{ij} &= \frac{\sum_{l=1}^{K}\sum_{t=1}^{T_l-1} P_\lambda(q_t{=}i\ q_{t+1}{=}j|o^l)}{\sum_{l=1}^{K}\sum_{t=1}^{T_l-1} P_\lambda(q_t{=}i|o^l)} \\
\bar{g}_{ik} &= \frac{\sum_{l=1}^{K}\sum_{t=1}^{T_l} P_\lambda(q_t{=}i\ m_t{=}k|o^l)}{\sum_{l=1}^{K}\sum_{t=1}^{T} P_\lambda(q_t{=}i|o^l)} \\
\bar{\mu}_{ikn} &= \frac{\sum_{l=1}^{K}\sum_{t=1}^{T_l} P_\lambda(q_t{=}i\ m_t{=}k|o^l)\ o^l_{tn}}{\sum_{t=1}^{T} P_\lambda(q_t{=}i|o^l)} \\
\bar{\sigma}^2_{ikn} &= \frac{\sum_{l=1}^{K}\sum_{t=1}^{T_l} P_\lambda(q_t{=}i\ m_t{=}k|o^l)\ (o_{tn}-\bar{\mu}_{ikn})^2}{\sum_{l=1}^{K}\sum_{t=1}^{T_l} P_\lambda(q_t{=}i|o^l)}
\end{aligned}
\tag{79}
$$

where $l = 1, ..., K$ indexes the training sequence, $t = 1, ..., T_l$ indexes the time within a training sequence, $i = 1, ..., N$ and $j = 1, ..., N$ index the hidden state, $k = 1, ..., M$ indexes the cluster embedded within a state, and $n = 1, ..., P$ indexes the dimension of the continuous vector of observations (e.g. the cepstral coefficient). The $P(q_t = jm_t = k|o)$, $P_\lambda(q_t = i|o^l)$ and $P_\lambda(q_t = i\ q_{t+1} = j|o^l)$ terms are obtained from the scaled forward-backward algorithms according to the following formulas:

$$
\begin{aligned}
P(q_t = i, m_t = k|o) &= P(q_t = i|o)\frac{g_{ik}\ \phi(o_t, \mu_{ik}, \Sigma_{ik})}{b_i(o_t)} \\
b_i(o_t) &= \sum_{k=1}^{M} g_{ik}\phi(o_t, \mu_{ik}, \Sigma_{ik}) \\
\phi(o_t, \mu_{ik}, \Sigma_{ik}) &= \frac{1}{(2\pi)^{P/2}\ \prod_{n=1}^{P}\sigma_{ikn}} e^{-d(o_t,\mu_{ik})} \\
d(o_t, \mu_{ik}) &= \frac{1}{2}\sum_{n=1}^{P}\left(\frac{o_{tn}-\mu_{ikn}}{\sigma_{ikn}}\right)^2 \\
P_\lambda(q_t = i|o^l) &= \frac{\hat{\alpha}^l_t(i)\hat{\beta}^l_t(i)}{\sum_i \hat{\alpha}^l_t(i)\hat{\beta}^l_t(i)} \\
P_\lambda(q_t = i\ q_{t+1} = j|o^l) &= \hat{\alpha}_t(i)a_{ij}\hat{\beta}_{t+1}(j)b_j(o^l_{t+1})
\end{aligned}
\tag{80}
$$

4.4 Viterbi decoding

We can use Viterbi decoding to find the best possible sequence of high level hidden states q and low level clusters m. There are two approaches to this problem. One approach attempts to find simultaneously the best joint sequence of high-level and low-level states. Thus the goal is to find $\hat{q}\hat{m} = \arg\max_{qm} P_\lambda(qm|o)$. The second approach first finds the best possible sequence of high level states $\hat{q} = \arg\max_q P_\lambda(q|o)$ and once $\hat{q}$ has been found, $\hat{m}$ is defined as $\hat{m} = \arg\max_m P_\lambda(\hat{m}|\hat{q}\ o)$. The two approaches do not necessarily yield the same results. The second approach is the standard in the literature.

For the second, most used version, of Viterbi decoding, the same Viterbi recurrence as in the discrete case applies but using the continuous version of $b_i(o_t)$. Once we have $\hat{q}_t$, the desired $\hat{m}_t$ is simply the cluster within $S_{\hat{q}_t}$ which is closest (in Mahalanobis distance) to o_t.

4.5 Viterbi training

The objective in Viterbi training (also known as segmental k-means) is to find an optimal point (local maximum) of $log\ \ P_\lambda(o\hat{q}\hat{m})$ with q and λ being the optimizing variables. It does not matter how $\hat{q}\hat{m}$ are found as long as a consistent procedure is used throughout training. As in the discrete case we define a dummy model $\hat{\lambda}$ such that $P_{\hat{\lambda}}(q^l\ m^l|o) = \delta(\hat{q}^l, q^l)\delta(\hat{m}^l, m^l)$. For such model, $P_{\hat{\lambda}}(q_1^l = j|o^l) = \delta(i, \hat{q}_1^l)$, $P_{\hat{\lambda}}(q_t^l = i|o^l) = \delta(i, \hat{q}_t^l)$, $P_{\hat{\lambda}}(q_t^l = i\ q_{t+1}^l = j|o^l) = \delta(i, \hat{q}_t^l)\delta(j, \hat{q}_{t+1}^l)$, and $P_{\hat{\lambda}}(q_t^l = i\ m_t^l = k|o^l) = \delta(i, \hat{q}_t^l)\delta(k, \hat{m}_t^l)$.

As in the discrete case note that $Q(\hat{\lambda}, \bar{\lambda}) = log\ \ P_{\bar{\lambda}}(o\ \hat{q}\hat{m})$, the function we want to optimize. Thus, substituting the $\hat{\lambda}$ parameters in the standard E-M formulas guarantees that $Q(\hat{\lambda}, \bar{\lambda}) \geq Q(\hat{\lambda}, \lambda)$, and thus $log\ P_{\bar{\lambda}}(o\ \hat{q}\hat{m}) \geq log\ P_\lambda(o\ \hat{q}\hat{m})$

Thus, the Viterbi training rules are as follows:

$$
\begin{aligned}
\bar{\pi}_i &= \frac{\sum_{l=1}^{K} \delta(i,\hat{q}_1^l)}{K} \\
\bar{a}_{ij} &= \frac{\sum_{l=1}^{K} \sum_{t=1}^{T_l-1} \delta(i,\hat{q}_t^l)\delta(j,\hat{q}_{t+1}^l)}{\sum_{l=1}^{K} \sum_{t=1}^{T_l-1} \delta(i,\hat{q}_t^l)} \\
\bar{g}_{ik} &= \frac{\sum_{l=1}^{K} \sum_{t=1}^{T_l} \delta(i,\hat{q}_t^l)\delta(k,\hat{m}_t^l)}{\sum_{l=1}^{K} \sum_{t=1}^{T} \delta(i,\hat{q}_t^l)} \\
\bar{\mu}_{ikn} &= \frac{\sum_{l=1}^{K} \sum_{t=1}^{T_l} \delta(i,\hat{q}_t^l)\delta(k,\hat{m}_t^l)\ o_{tn}^l}{\sum_{t=1}^{T} \delta(i,\hat{q}_t^l)} \\
\bar{\sigma^2}_{ikn} &= \frac{\sum_{l=1}^{K} \sum_{t=1}^{T_l} \delta(i,\hat{q}_t^l)\delta(k,\hat{m}_t^l)(o_{tn}-\bar{\mu}_{ikn})^2}{\sum_{l=1}^{K} \sum_{t=1}^{T_l} \delta(i,\hat{q}_t^l)}
\end{aligned}
\tag{81}
$$

Since Viterbi decoding maximizes $P_\lambda(q, m, o)$ with respect to q and m and Viterbi training maximizes $P_\lambda(q, m, o)$ with respect to λ, repeatedly applying Viterbi decoding and

Viterbi training, can only make $P_\lambda(q, m, o)$ increase and convergence to a local maximum is guaranteed.

5 Factored Sampling Methods for Continuous State Models

Many recognition problems can be framed in terms of infering something about q_t the internal state of a system, based on a sequence of observations $o = o_1 \cdots o_t$. These inferences are in many cases based on estimates of $p(q_t|o_1 \cdots o_t)$. When the states are discrete and countable, these conditonal state probabilities can be obtained using the forwards algorithm. However, the algorithm cannot be used when the states are continuous. In such case, direct sampling methods are appropriate. Here is an example of how these methods work. We start with a sensor model: $p(o_t|q_t)$ and a Markovian state dynamics model $p(q_{t+1}|q_t)$. Our goal is to obtain estimates of $p(q_t|\underline{o}_t)$ for all t.

1. **Recursion**
   Assume we have an estimate $\hat{p}(q_t|\underline{o}_t)$. Our goal is to update that estimate for the next time step $p(q_{t+1}|\underline{o}_{t+1})$.
   (a) First we draw a random sample X from $\hat{p}(q_t|\underline{o}_t)$. This sample will implicitely define a re-estimation of $p(q_t|\underline{o}_t)$ in terms of a mixture of delta functions: $\hat{p}(q_t|\underline{o}_t) = \frac{1}{N}\sum_{i=1}^{N}\delta(q_t, x_i)$
   (b) For each observation x_i we obtain another random observation y_i using the state dynamics $p(q_{t+1}|q_t = x_i)$. The new sample $Y = \{y_1 \cdots y_n\}$ implicitely defines our estimates of $p(q_{t+1}|\underline{o}_t)$.
   $p(q_t|\underline{o}_t) = \frac{1}{N}\sum_{i=1}^{N}\delta(q_{t+1}, y_i)$
   (c) We know that

$$p(q_{t+1}|o_1 \cdots o_{t+1}) = \frac{p(o_1 \cdots o_t)}{p(o_1 \cdots o_{t+1})} p(q_{t+1}o_{t+1}|o_1 \cdots o_t) = \frac{p(o_1 \cdots o_t)}{p(o_1 \cdots o_{t+1})} p(q_{t+1}|o_1 \cdots o_t) p(o_{t+1}|q_{t+1}) \tag{82}$$

   The fraction is a constant $K(\underline{o}_{t+1})$ independent of q_{t+1}, we already have an estimate of $p(q_{t+1}|\underline{o}_t)$ so we just need to weight it by $p(o_{t+1}|q_{t+1})$.
   $\hat{p}(q_{t+1}|\underline{o}_{t+1}) = K\ \frac{1}{N}\sum_{i=1}^{N}\delta(q_{t+1}, y_i)p(o_{t+1}|q_{t+1}) =$

$$= \frac{1}{(N)\sum_i p(o_{t+1}|y_i)} \sum_{i=1}^{N} \delta(q_{t+1}, y_i) p(o_{t+1}|y_i)$$

   We can now use $\hat{p}(q_{t+1}|\underline{o}_{t+1})$ to estimate parameters like the mean or the variance of the distribution. More generally,
   $\hat{\omega} = \int dq_{t+1} Q(p(q_{t+1}|\underline{o}_{t+1}), q_{t+1})$
2. **Initialization**
   The initialization step is basically the same as the recursion step only that instead of using the state transition probabilities we use the initial state probabilities
   (a) Obtain a sample of N random states : X= $\{x_1 \cdots x_N\}$ from the initial state probability function $\pi(\cdot)$. These N samples will implicitely work as our estimate of the initial state probability.

$$\hat{p}(q_1) = \frac{1}{N}\sum_{i=1}^{N}\delta(q_1, y_i) \tag{83}$$

   (b) Weight each observation by the sensor probability $p(o_1|q_1 = y_i)$. This defines our initial distribution estimates

$$\hat{p}(q_0|o_1) = \frac{1}{N\sum_{i=1}^{N} p(o_1|x_i)} \delta(q_0, y_i) p(o_0|y_i) \tag{84}$$

6 History

- The first version of this document was written by Javier R. Movellan and it was 18 pages long.
- The document was made open source under GDPL license as part of the Kolmogorov project on August 10, 2002.
- October 9, 2003. Javier R. Movellan changed the license to GFDL 1.2 and included an endorsement section.