A Simple Markov Model For Weather Prediction

What is a first-order Markov chain?

$$P[q_t = j | (q_{t-1} = i, q_{t-2} = k, \ldots)] = P[q_t = j | q_{t-1} = i]$$

We consider only those processes for which the right-hand side is independent of time:

$$a_{ij} = P[q_t = j | q_{t-1} = i] \qquad 1 \le i, j \le N$$

with the following properties:

$$a_{ij} \ge 0 \qquad \forall j, i$$

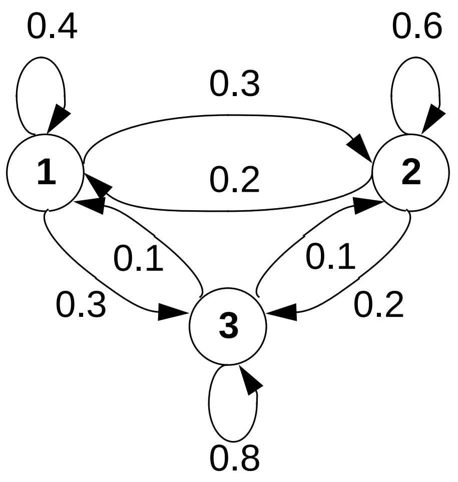
$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \forall i$$

The above process can be considered observable because the output process is a set of states at each instant of time, where each state corresponds to an observable event.

Later, we will relax this constraint, and make the output related to the states by a second random process.

Example: A three-state model of the weather

State 1: precipitation (rain, snow, hail, etc.)
State 2: cloudy
State 3: sunny



Basic Calculations

Example: What is the probability that the weather for eight consecutive days is "sun-sun-sun-rain-rain-sun-cloudy-sun"?

Solution:

O =	sun	sun	sun	rain	rain	sun	cloudy	sun
	3	3	3	1	1	3	2	3

$$\begin{aligned} P(\bar{O}|Model) &= P[3]P[3|3]P[3|3]P[1|3]P[1|1]P[3|1]P[2|3]P[3|2] \\ &= \pi_3 a_{33} a_{31} a_{11} a_{13} a_{32} a_{23} \\ &= 1.536 \times 10^{-4} \end{aligned}$$

Example: Given that the system is in a known state, what is the probability that it stays in that state for *d* days?

O = i i i ... i j

$$\begin{aligned} P(\bar{O}|Model, q_1 = i) &= P(\bar{O}, q_1 = i|Model)/P(q_1 = i) \\ &= \pi_i a_{ii}^{d-1}(1 - a_{ii})/\pi_i \\ &= a_{ii}^{d-1}(1 - a_{ii}) \\ &= p_i(d) \end{aligned}$$

Note the exponential character of this distribution.

We can compute the expected number of observations in a state given that we started in that state:

$$\bar{d}_i = \sum_{d=1}^{\infty} d p_i(d) = \sum_{d=1}^{\infty} d a_{ii}^{d-1}(1 - a_{ii}) = \frac{1}{1 - a_{ii}}$$

Thus, the expected number of consecutive sunny days is (1/(1-0.8)) = 5; the expected number of cloudy days is 2.5, etc.

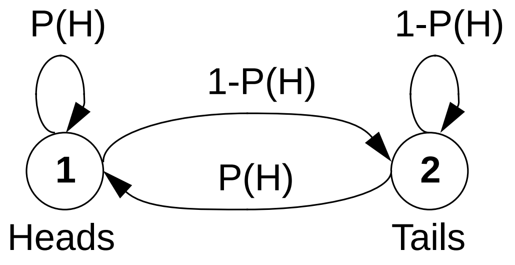
What have we learned from this example?

Why Are They Called "Hidden" Markov Models?

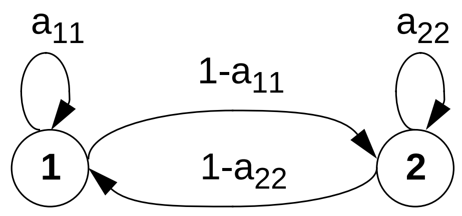
Consider the problem of predicting the outcome of a coin toss experiment. You observe the following sequence:

$$\bar{O} = (HHTTTHTTH\ldots H)$$

What is a reasonable model of the system?

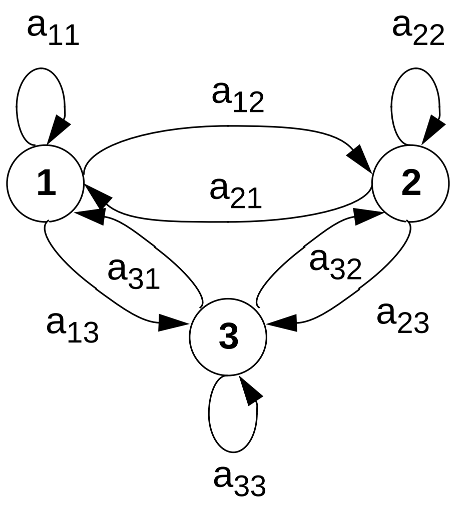


1-Coin Model
(Observable Markov Model)
O = H H T T H T H H T T H ...
S = 1 1 2 2 1 2 1 1 2 2 1 ...



2-Coins Model
(Hidden Markov Model)
O = H H T T H T H H T T H ...
S = 1 1 2 2 1 2 1 1 2 2 1 ...

$P(H) = P_1$ $P(H) = P_2$
$P(T) = 1-P_1$ $P(T) = 1-P_2$

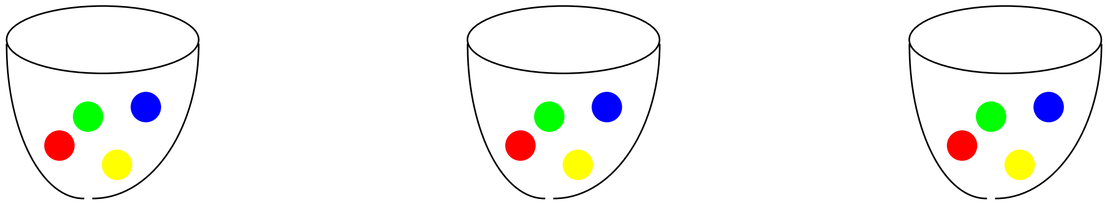


3-Coins Model
(Hidden Markov Model)
O = H H T T H T H H T T H ...
S = 3 1 2 3 3 1 1 2 3 1 3 ...

P(H):	P_1	P_2	P_3
P(T):	$1-P_1$	$1-P_2$	$1-P_3$

Why Are They Called Doubly Stochastic Systems?

<u>The Urn-and-Ball Model</u>



P(red)	$= b_1(1)$	P(red)	$= b_2(1)$	P(red)	$= b_3(1)$
P(green)	$= b_1(2)$	P(green)	$= b_2(2)$	P(green)	$= b_3(2)$
P(blue)	$= b_1(3)$	P(blue)	$= b_2(3)$	P(blue)	$= b_3(3)$
P(yellow)	$= b_1(4)$	P(yellow)	$= b_2(4)$	P(yellow)	$= b_3(4)$
...		...		...	

$$\overline{O} = \{\text{green, blue, green, yellow, red, ..., blue}\}$$

How can we determine the appropriate model for the observation sequence given the system above?

Elements of a Hidden Markov Model (HMM)

- N — the number of states
- M — the number of distinct observations per state
- The state-transition probability distribution $\underline{A} = \{a_{ij}\}$
- The output probability distribution $\underline{B} = \{b_j(k)\}$
- The initial state distribution $\pi = \{\pi_i\}$

We can write this succinctly as: $\lambda = (\underline{A}, \underline{B}, \pi)$

Note that the probability of being in any state at any time is completely determined by knowing the initial state and the transition probabilities:

$$\pi(t) = \underline{A}^{t-1}\pi$$

Two basic problems:

(1) how do we train the system?

(2) how do we estimate the probability of a given sequence (recognition)?

This gives rise to a third problem:

If the states are hidden, how do we know what states were used to generate a given output?

How do we represent continuous distributions (such as feature vectors)?

Formalities

The *discrete observation* HMM is restricted to the production of a finite set of discrete observations (or sequences). The output distribution at any state is given by:

$$b(k, i) \equiv P(\underline{y}(t) = k | \underline{x}(t) = i)$$

The observation probabilities are assumed to be independent of time. We can write the probability of observing a particular observation, $\underline{y}(t)$, as:

$$b(y(t)|i) \equiv P(\underline{y}(t) = y(t) | \underline{x}(t) = i)$$

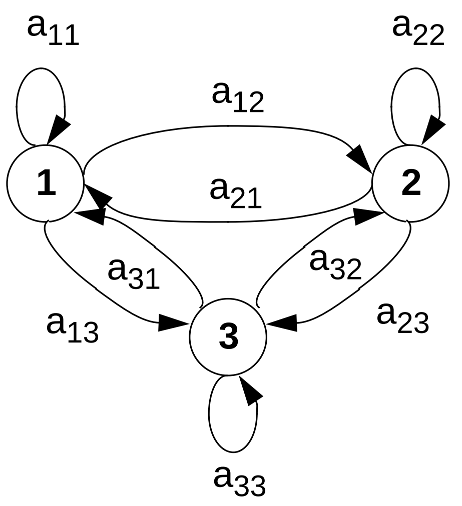
The observation probability distribution can be represented as a matrix whose dimension is K rows x S states.
We can define the observation probability vector as:

$$\boldsymbol{p}(t) = \begin{bmatrix} P(\underline{y}(t) = 1) \\ P(\underline{y}(t) = 2) \\ \dots \\ P(\underline{y}(t) = K) \end{bmatrix}, \quad \text{or,} \quad \boldsymbol{p}(t) = \boldsymbol{B}\pi(t) = \boldsymbol{B}\boldsymbol{A}^{t-1}\pi(1)$$

The mathematical specification of an HMM can be summarized as:

$$\boldsymbol{M} = \{S, \pi(1), \boldsymbol{A}, \boldsymbol{B}, \{\boldsymbol{y}_k, 1 \le k \le K\}\}$$

For example, reviewing our coin-toss model:



P(H):	P_1	P_2	P_3
P(T):	$1\text{-}P_1$	$1\text{-}P_2$	$1\text{-}P_3$

$$S = 3$$

$$\pi(1) = \begin{Bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{Bmatrix}$$

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} P_1 & P_2 & P_3 \\ 1 - P_1 & 1 - P_2 & 1 - P_3 \end{bmatrix}$$

Recognition Using Discrete HMMs

Denote any partial sequence of observations in time by:

$$y_{t_1}^{t_2} \equiv \{y(t_1), y(t_1+1), y(t_1+2), \ldots, y(t_2)\}$$

The forward partial sequence of observations at time t is

$$y_1^t \equiv \{y(1), y(2), \ldots, y(t)\}$$

The backward partial sequence of observations at time t is

$$y_{t+1}^T \equiv \{y(t+1), y(t+2), \ldots, y(T)\}$$

A complete set of observations of length T is denoted as $y \equiv y_1^T$.

What is the likelihood of an HMM?

We would like to calculate $P(M | \underline{y} = y)$ — however, we can't. We can (see the introductory notes) calculate $P(\underline{y} = y | M)$. Consider the brute force method of computing this. Let $\vartheta = \{i_1, i_2, \ldots, i_T\}$ denote a specific state sequence. The probability of a given observation sequence being produced by this state sequence is:

$$P(y | \vartheta, M) = b(y(1) | i_1) b(y(2) | i_2) \ldots b(y(T) | i_T)$$

The probability of the state sequence is

$$P(\vartheta | M) = P(\underline{x}(1) = i_1) a(i_2 | i_1) a(i_3 | i_2) \ldots a(i_T | i_{T-1})$$

Therefore,

$$\begin{aligned} P(y, (\vartheta | M)) = {} & P(\underline{x}(1) = i_1) a(i_2 | i_1) a(i_3 | i_2) \ldots a(i_T | i_{T-1}) \\ & x \;\; b(y(1) | i_1) b(y(2) | i_2) \ldots b(y(T) | i_T) \end{aligned}$$

To find $P(y | M)$, we must sum over all possible paths:

$$P(y | M) = \sum_{\forall \vartheta} P(y, (\vartheta | M))$$

This requires $O(2TS^T)$ flops. For $S = 5$ and $T = 100$, this gives about 1.6×10^{72} computations per HMM!

The "Any Path" Method (Forward-Backward, Baum-Welch)

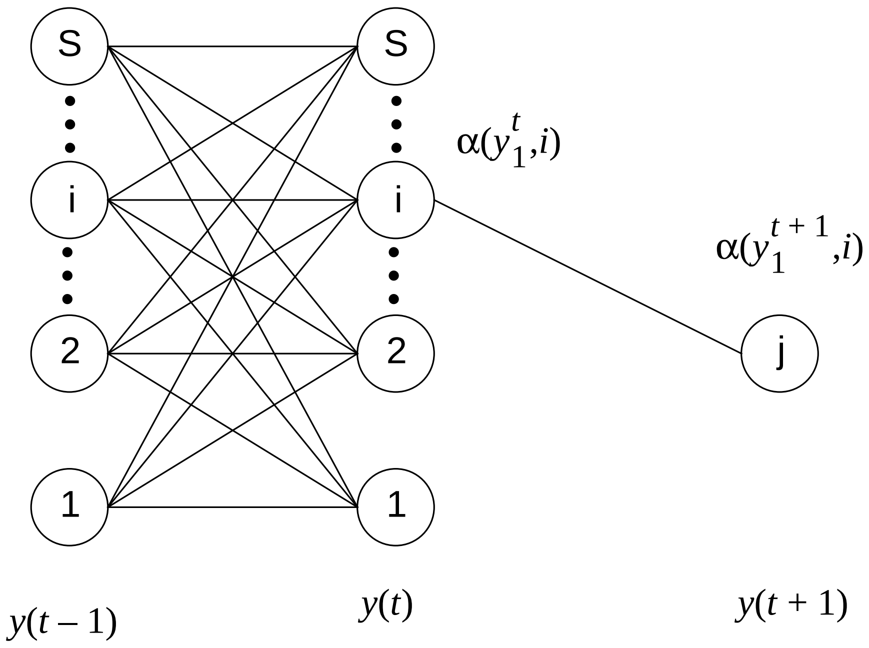
The *forward-backward* (F-B) *algorithm* begins by defining a "forward-going" probability sequence:

$$\alpha(y_1^t) \equiv P(\underline{y}_1^t = y_1^t, \underline{x}(t) = i \,|\, M)$$

and a "backward-going" probability sequence:

$$\beta(y_{t+1}^T \,|\, i) \equiv P(\underline{y}_{t+1}^T = y_{t+1}^T \,|\, \underline{x}(t) = i, M)$$

Let us next consider the contribution to the overall sequence probability made by a single transition:



$$\begin{aligned}\alpha(y_1^{t+1}, j) &= \alpha(y_1^t, i) P(\underline{x}(t+1) = j \,|\, \underline{x}(t) = i) \times \\ &\quad P(\underline{y}(t+1) = y(t+1) \,|\, \underline{x}(t+1) = j) \\ &= \alpha(y_1^t, i) a(j|i) b(y(t+1)|j)\end{aligned}$$

Summing over all possibilities for reaching state "j":

$$\alpha(y_1^{t+1}, j) = \sum_{i=1}^{S} \alpha(y_1^t, i) a(j|i) b(y(t+1)|j)$$

Baum-Welch (Continued)

The recursion is initiated by setting:

$$\alpha(y_1^t, j) = P(\underline{x}(1) = j) b(y(1) | j)$$

Similarly, we can derive an expression for β:

$$\beta(y_{i+1}^T | i) = \sum_{j=1}^{S} \beta(y_{t+2}^T | j) a(j|i) b(y(t+1) | j)$$

This recursion is initialized by:

$$\beta(y_{T+1}^T | i) \equiv \begin{cases} 1, & \text{if } i \text{ is a legal final state} \\ 0, & \text{otherwise} \end{cases}$$

We still need to find $P(y|M)$:

$$P(y, \underline{x}(t) = i | M) = \alpha(y_1^t, i) \beta(y_{t+1}^T | i)$$

for any state i. Therefore,

$$P(y|M) = \sum_{i=1}^{S} \alpha(y_1^t, i) \beta(y_{t+1}^T | i)$$

But we also note that we should be able to compute this probability using only the forward direction. By considering $t = T$, we can write:

$$P(y|M) = \sum_{i=1}^{S} \alpha(y_1^T, i)$$

These equations suggest a recursion in which, for each value of t we iterate over ALL states and update $\alpha(y_1^t, j)$. When $t = T$, $P(y|M)$ is computed by summing over ALL states.

The complexity of this algorithm is $O(S^2 T)$, or for $S = 5$ and $T = 100$, approximately 2500 flops are required (compared to 10^{72} flops for the exhaustive search).

The Viterbi Algorithm

Instead of allowing any path to produce the output sequence, and hence, creating the need to sum over all paths, we can simply assume only one path produced the output. We would like to find the single most likely path that could have produced the output. Calculation of this path and probability is straightforward, using the dynamic programming algorithm previously discussed:

$$D(t, i) = a(i, j^*)b(k|i)D(t-1, j^*)$$

where

$$j^* = \underset{valid\ j}{\arg max}\{D(t-1, j)\}$$

(in other words, the predecessor node with the best score). Often, probabilities are replaced with the logarithm of the probability, which converts multiplications to summations. In this case, the HMM looks remarkably similar to our familiar DP systems.

Beam Search

In the context of the best path method, it is easy to see that we can employ a beam search similar to what we used in DP systems:

$$D_{min}(t, i) \ge D_{min}(t, i^*_t) - \delta(t)$$

In other words, for a path to survive, its score must be within a range of the best current score. This can be viewed as a time-synchronous beam search. It has the advantage that, since all hypotheses are at the same point in time, their scores can be compared directly. This is due to the fact that each hypothesis accounts for the same amount of time (same number of frames).

Training Discrete Observation HMMs

Training refers to the problem of finding $\{\pi(1), \boldsymbol{A}, \boldsymbol{B}\}$ such that the model, M, after an iteration of training, better represents the training data than the previous model. The number of states is usually not varied or reestimated, other than via the modification of the model inventory. The apriori probabilities of the likelihood of a model, $\pi(1)$, are normally not reestimated as well, since these typically come from the language model.

The first algorithm we will discuss is one based on the **Forward-Backward** algorithm (Baum-Welch Reestimation):

$$u_{j|i} \equiv \text{label for a transition from state } i \text{ to state } j$$

$$u_{\bullet|i} \equiv \text{set of transitions exiting state } i$$

$$u_{j|\bullet} \equiv \text{set of transitions entering } j$$

Also, $\underline{u}(t)$ denotes a random variable that models the transitions at time t and $\underline{y}_j(t)$ a random variable that models the observation being emitted at state j at time t. The symbol "•" is used to denote an arbitrary event.

Next, we need to define some intermediate quantities related to particular events at a given state at a given time:

$$\begin{aligned}\zeta(i, j; t) &\equiv P(\underline{u}(t) = u_{j|i} \mid y, M) \\ &= P(\underline{u}(t) = u_{j|i}, y \mid M) / P(y|M) \\ &= \left\{\begin{array}{ll} \dfrac{\alpha(y_1^t, i) a(j|i) b(y(t+1)|j) \beta(y_{t+2}^T \mid j)}{P(y|M)}, & t = 1, 2, \ldots, T \\ 0\,, & \text{other } t \end{array}\right\}\end{aligned}$$

where the sequences α, β, a, and b were defined previously (last lecture). Intuitively, we can think of this as the probability of observing a transition from state i to state j at time t for a particular observation sequence, y, (the utterance in progress), and model M.

We can also make the following definition:

$$\gamma(i;t) \equiv P(\underline{u}(t) \in u_{\bullet|i} \,|\, y, M) = \sum_{j=1}^{S} \zeta(i, j; t)$$

$$= \begin{cases} \dfrac{\alpha(y_1^t, i)\beta(y_{t+1}^T | i)}{P(y|M)}, & t = 1, 2, \ldots, T \\ 0, & \text{other } t \end{cases}$$

This is the probability of exiting state i. Also,

$$\nu(j;t) \equiv P(\underline{x}(t) = j | y, M)$$

$$= \begin{cases} \gamma(j;t), & t = 1, 2, \ldots, T \\ \alpha(y_1^T, j), & t = T \\ 0, & \text{other } t \end{cases}$$

$$= \begin{cases} \dfrac{\alpha(y_1^t, j)\beta(y_{t+1}^T | j)}{P(y|M)}, & t = 1, 2, \ldots, T \\ 0, & \text{other } t \end{cases}$$

which is the probability of being in state j at time t. Finally,

$$\delta(j,k;t) \equiv P(\underline{y}_j(t) = k | y, M)$$

$$= \begin{cases} \nu(j;t), & \text{if } y(t) = k \text{ and } 1 \le t \le T \\ 0, & \text{otherwise} \end{cases}$$

$$= \begin{cases} \dfrac{\alpha(y_1^t, j)\beta(y_{t+1}^T | j)}{P(y|M)}, & \text{if } y(t) = k \text{ and } 1 \le t \le T \\ 0, & \text{otherwise} \end{cases}$$

which is the probability of observing symbol k at state j at time t.

Note that we make extensive use of the forward and backward probabilities in these computations. This will be key to reducing the complexity of the computations by allowing an interactive computation.

From these four quantities, we can define four more intermediate quantities:

$$\zeta(i, j;\bullet) = P(\underline{u}(\bullet) \in u_{j|i} | y, M) = \sum_{t=1}^{T} \zeta(i, j;t)$$

$$\gamma(i;\bullet) = P(\underline{u}(\bullet) \in u_{\bullet|i} | y, M) = \sum_{t=1}^{T} \gamma(i;t)$$

$$\nu(j;\bullet) = P(\underline{u}(\bullet) \in u_{j|\bullet} | y, M) = \sum_{t=1}^{T} \nu(j;t)$$

$$\delta(j, k;\bullet) = P(\underline{y}_j(\bullet) = k | y, M) = \sum_{t=1}^{T} \delta(j, k;t) = \sum_{\substack{t = 1 \\ y(t) = k}}^{T} \nu(j;t)$$

Finally, we can begin relating these quantities to the problem of reestimating the model parameters. Let us define four more random variables:

$$\underline{n}(u_{j|i}) \equiv \text{number of transitions of the type } u_{j|i}$$

$$\underline{n}(u_{\bullet|i}) \equiv \text{number of transitions of the type } u_{\bullet|i}$$

$$\underline{n}(u_{j|\bullet}) \equiv \text{number of transitions of the type } u_{j|\bullet}$$

$$\underline{n}(\underline{y}_j(\bullet) = k) \equiv \text{number of times the observation } k \text{ and state } j \text{ jointly occur}$$

We can see that:

$$\zeta(i, j;\bullet) = E\{\underline{n}(u_{j|i}) | y, M\}$$

$$\gamma(i;\bullet) = E\{\underline{n}(u_{\bullet|i}) | y, M\}$$

$$\nu(j;\bullet) = E\{\underline{n}(u_{j|\bullet}) | y, M\}$$

$$\delta(j, k;\bullet) = E\{\underline{n}(\underline{y}_j(\bullet) = k) | y, M\}$$

What we have done up to this point is to develop expressions for the estimates of the underlying components of the model parameters in terms of the state sequences that occur during training.

But how can this be when the internal structure of the model is **hidden**?

Following this line of reasoning, an estimate of the transition probability is:

$$\bar{a}(j|i) = \frac{E\{\underline{n}(u_{j|i})|y,M\}}{E\{\underline{n}(u_{\bullet|i})|y,M\}} = \frac{\zeta(i,j;\bullet)}{\gamma(i;\bullet)}$$

$$= \frac{\sum_{t=1}^{T-1} \alpha(y_1^t,i)a(j|i)b(y(t+1)|j)\beta(y_{t+2}^T|j)}{\sum_{t=1}^{T-1} \alpha(y_1^t,i)\beta(y_{t+1}^T|i)}$$

Similarly,

$$\bar{b}(k|j) = \frac{E\left\{\underline{n}(\underline{n}(\underline{y}_j(\bullet)=k)|y,M)|y,M\right\}}{E\{\underline{n}(u_{j|\bullet})|y,M\}} = \frac{\zeta(i,j;\bullet)}{\gamma(i;\bullet)}$$

$$= \frac{\sum_{\substack{t=1 \\ y(t)=k}}^{T} \alpha(y_1^t,j)\beta(y_t^T|j)}{\sum_{t=1}^{T} \alpha(y_1^t,j)\beta(y_{t+1}^T|j)}$$

Finally,

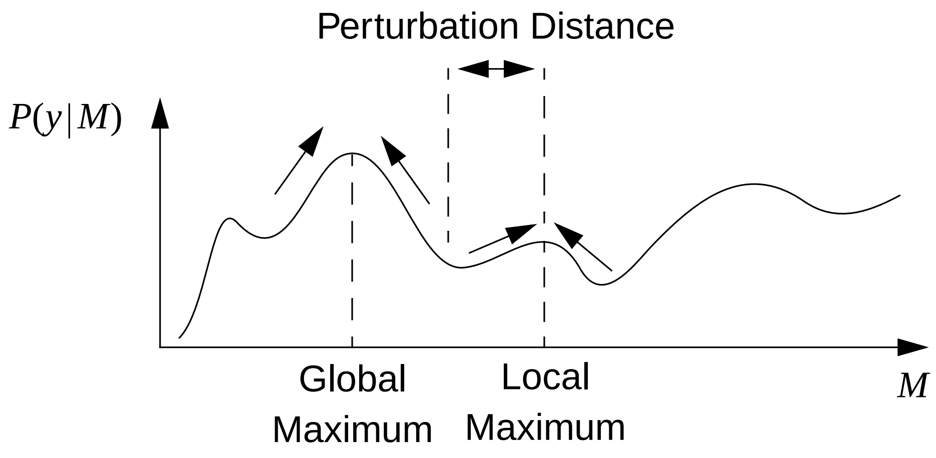
$$P(\underline{x}(1)=i) = \frac{\alpha(y_1^1,i)\beta(y_2^T|i)}{P(y|M)}$$

This process is often called reestimation by recognition, because we need to recognize the input with the previous models in order that we can compute the new model parameters from the set of state sequences used to recognize the data (hence, the need to iterate).

But will it converge? Baum and his colleagues showed that the new model guarantees that:

$$P(y|\overline{M}) \ge P(y|M)$$

Since this is a highly nonlinear optimization, it can get stuck in local minima:



We can overcome this by starting training from a different initial point, or "bootstrapping" models from previous models.

Analogous procedures exist for the **Viterbi algorithm**, though they are much simpler and more intuitive (and more DP-like):

$$\bar{a}(j|i) = \frac{E\{\underline{n}(u_{j|i})\big|y,M\}}{E\{\underline{n}(u_{\bullet|i})\big|y,M\}}$$

and,

$$\bar{b}(k|j) = \frac{E\{\underline{n}(u_{j|i})\big|y,M\}}{E\{\underline{n}(u_{\bullet|i})\big|y,M\}}$$

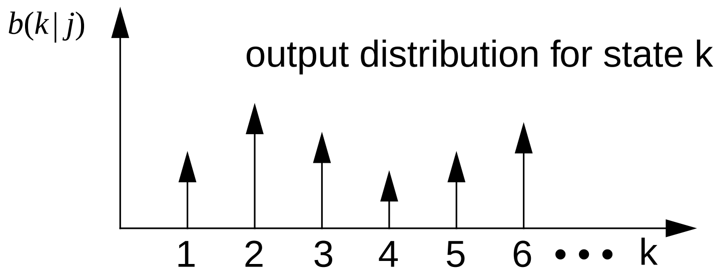
These have been shown to give comparable performance to the forward-backward algorithm at significantly reduced computation. It also is generalizable to alternate formulations of the topology of the acoustic model (or language model) drawn from formal language theory. (In fact, we can even eliminate the first-order Markovian assumption.)

Further, the above algorithms are easily applied to many problems associated with language modeling: estimating transition probabilities and word probabilities, efficient parsing, and learning hidden structure.

But what if a transition is never observed in the training database?

Continuous Density HMMs

The discrete HMM incorporates a discrete probability density function, captured in the matrix $\boldsymbol{B}$, to describe the probability of outputting a symbol:



Signal measurements, or feature vectors, are continuous-valued N-dimensional vectors. In order to use our discrete HMM technology, we must vector quantize (VQ) this data — reduce the continuous-valued vectors to discrete values chosen from a set of M codebook vectors. Initially, most HMMs were based on VQ front-ends. However, recently, the continuous density model has become widely accepted.

Let us assume a parametric model of the observation pdf:

$$M = \left\{ S, \pi(1), \boldsymbol{A}, \left\{ f_{\underline{y}|\underline{x}}(\xi | i), 1 \le i \le S \right\} \right\}$$

The likelihood of generating observation $\boldsymbol{y}(t)$ in state j is defined as:

$$b(\boldsymbol{y}(t) | j) \equiv f_{\underline{y}|\underline{x}}(\boldsymbol{y}(t) | j)$$

Note that taking the negative logarithm of $b(\ )$ will produce a log-likelihood, or a Mahalanobis-like distance. But what form should we choose for $f(\ )$? Let's assume a Gaussian model, of course:

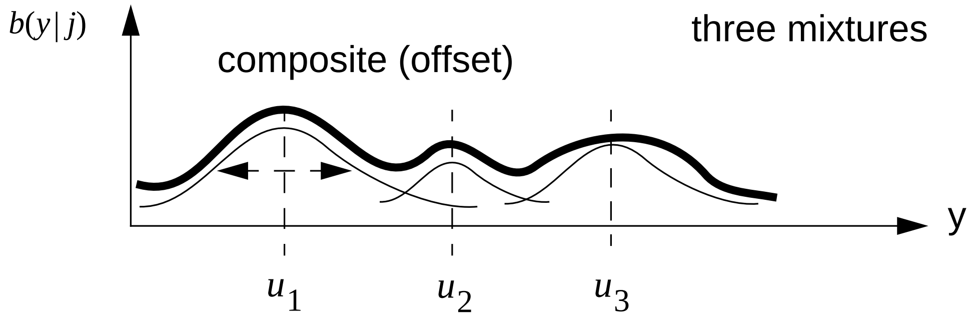
$$f_{\underline{y}|\underline{x}}(\boldsymbol{y} | i) = \frac{1}{\sqrt{2\pi |\boldsymbol{C}_i|}} \exp\left\{ -\frac{1}{2} (\boldsymbol{y} - \mu_i)^T \boldsymbol{C}_i^{-1} (\boldsymbol{y} - \mu_i) \right\}$$

Note that this amounts to assigning a mean and covariance matrix to each state — a significant increase in complexity. However, shortcuts such as variance-weighting can help reduce complexity.

Also, note that the log of the output probability at each state becomes precisely the Mahalanobis distance (principal components) we studied at the beginning of the course.

Mixture Distributions

Of course, the output distribution need not be Gaussian, or can be multimodal to reflect the fact that several contexts are being encoded into a single state (male/female, allophonic variations of a phoneme, etc.). Much like a VQ approach can model any discrete distribution, we can use a weighted linear combination of Gaussians, or a mixture distribution, to achieve a more complex statistical model.



Mathematically, this is expressed as:

$$f_{\underline{y}|\underline{x}}(\boldsymbol{y}|i) = \sum_{m=1}^{M} c_{im} \aleph(\boldsymbol{y};\mu_{im},\boldsymbol{C}_{im})$$

In order for this to be a valid pdf, the mixture coefficients must be nonnegative and satisfy the constraint:

$$\sum_{m=1}^{M} c_{im} = 1, \qquad 1 \le i \le S$$

Note that mixture distributions add significant complexity to the system: m means and covariances at each state.

Analogous reestimation formulae can be derived by defining the intermediate quantity:

$$\nu(i;t,l) \equiv P(\underline{x}(t) = i | \boldsymbol{y}(t) \text{produced in accordance with mixture } l)$$

$$= \frac{\alpha(\boldsymbol{y}_1^t, i)\beta(\boldsymbol{y}_{t+1}^T | i)}{\sum_{j=1}^{S} \alpha(\boldsymbol{y}_1^t, j)\beta(\boldsymbol{y}_{t+1}^T | j)} \times \frac{c_{il} \aleph(\boldsymbol{y}_{tl}^t;\mu_{il},\boldsymbol{C}_{il})}{\sum_{m=1}^{M} c_{im} \aleph(\boldsymbol{y}_t^t;\mu_{im},\boldsymbol{C}_{im})}$$

The mixture coefficients can now be reestimated using:

$$\bar{c}_{il} = \frac{\nu(i;\bullet,l)}{\sum_{m=1}^{M} \nu(i;\bullet,m)}$$

the mean vectors can be reestimated as:

$$\bar{\mu}_{il} = \frac{\sum_{t=1}^{T} \nu(i;t,l)\boldsymbol{y}(t)}{\nu(i;\bullet,l)}$$

the covariance matrices can be reestimated as:

$$\bar{\boldsymbol{C}}_{il} = \frac{\sum_{t=1}^{T} \nu(i;t,l)[\boldsymbol{y}(t)-\mu_{il}][\boldsymbol{y}(t)-\mu_{il}]^T}{\nu(i;\bullet,l)}$$

and the transition probabilities, and initial probabilities are reestimated as usual.

The Viterbi procedure once again has a simpler interpretation:

$$\mu_{il} = \frac{1}{N_{il}} \sum_{\substack{t=1 \\ y(t)\sim il}}^{T} \boldsymbol{y}(t)$$

and

$$\boldsymbol{C}_{il} = \frac{1}{N_{il}} \sum_{\substack{t=1 \\ y(t)\sim il}}^{T} [\boldsymbol{y}(t)-\mu_{il}][\boldsymbol{y}(t)-\mu_{il}]^T$$

The mixture coefficient is reestimated as the number of vectors associated with a given mixture at a given state:

$$c_{il} = \frac{N_{il}}{N_i}$$