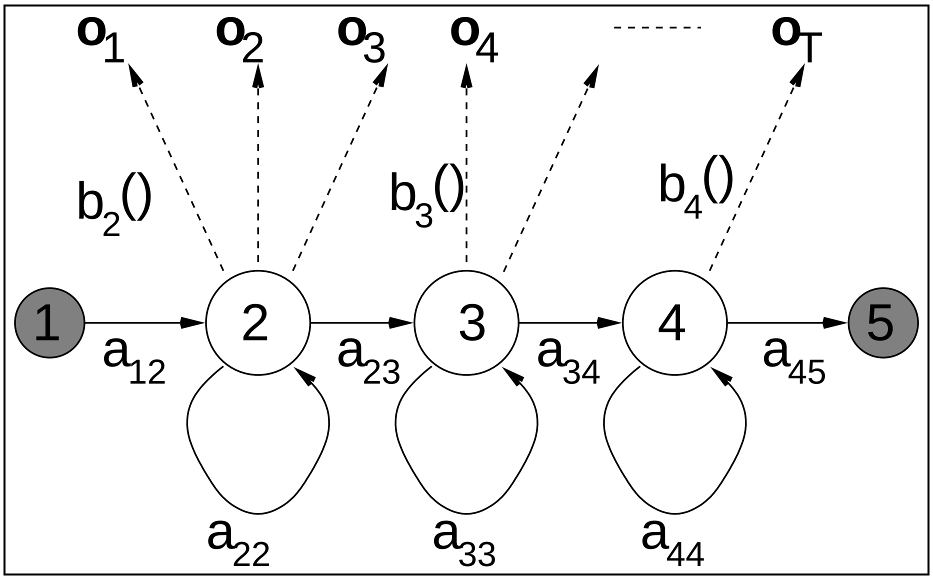
University of Cambridge

MPhil in Computer Speech Text & Internet Technology

Module: Speech Processing II

Lecture 2: Hidden Markov Models I



Lent 2003

Hidden Markov Models I & II

Review of basic HMMs

- HMM assumptions
- HMM structure and operation
- maximum likelihood training
- forward-backward algorithm

Output Probability Distributions

- continuous density
- mixture distributions
- discrete density & VQ
- semi-continuous (tied mixtures)

Duration Modelling

HMM Assumptions

Hidden Markov models (HMMs) are the most widely used acoustic model for speech recognition.

Assumptions:

1. The observations accurately represent the signal. Speech is assumed to be stationary over the length of the frame. Frames are usually around 25msecs, so this is not a bad assumption.
2. Observations are independent given the state that generated it. Previous and following observations do not affect the likelihood. This is not true for speech, speech has a high degree of continuity.
3. Between state transition probabilities are constant. The probability of from one state to another is independent of the observations and previously visited states. This is not a good model for speech.

Despite its limitations HMMs are, to date, the most successful acoustic models for speech recognition.

Neural Networks and hybrid HMM/neural network systems are also used (this will be discussed later in this module).

HMM Structure & Operation

An HMM is a finite state machine:

1. it has a variable number $N-2$ of *emitting states*.
2. it has a non-emitting *entry* state and a non-emitting *exit* state (some formulations don't explicitly include these).
3. any pair of states i and j can be connected by a transition with probability a_{ij}
4. it changes from current state i to new state j with probability a_{ij} every input frame
5. every time a state j is entered, an acoustic feature vector $\mathbf{o}$ is generated with probability $b_j(\mathbf{o})$

Note: the HMM is a *generative* model of speech. It can also be used to find the likelihood that the model generated the observed data.

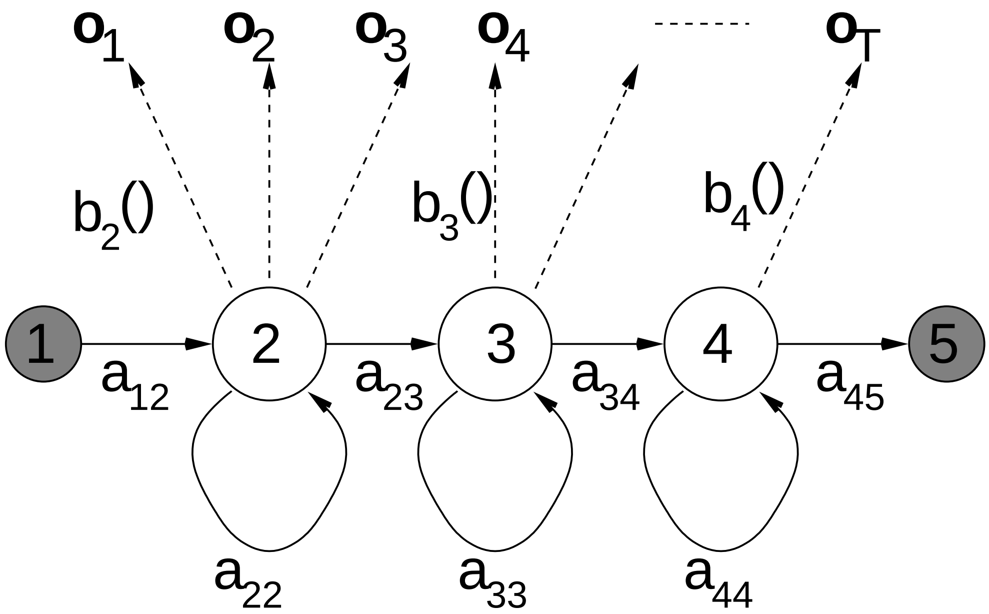


Figure 1:

For the HMM shown the output/state sequence is

time	1	2	3	4	...	T
observations	$\mathbf{o}_1$	$\mathbf{o}_2$	$\mathbf{o}_3$	$\mathbf{o}_4$	...	$\mathbf{o}_T$
states	2	2	2	3	...	4

The HMM parameters are:

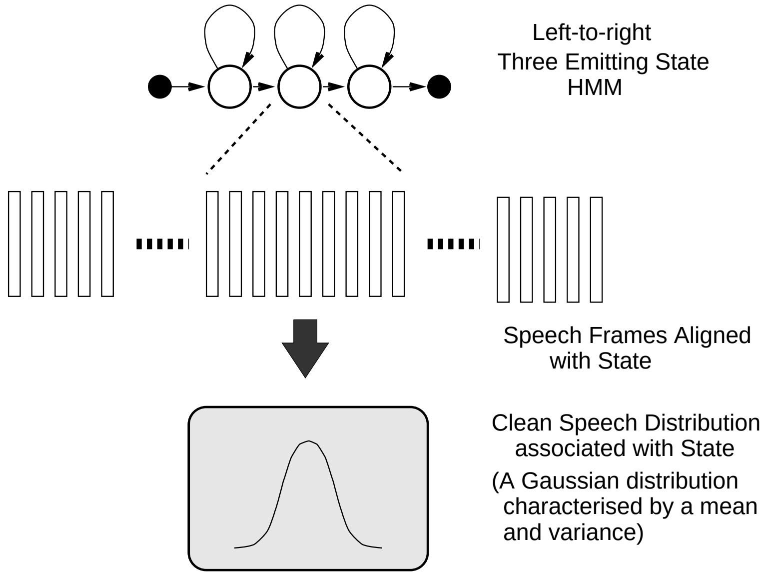
1. a_{ij}: The transition probability from state i to state j.
2. The parameters of the output probability distribution associated with each state. Thus $b_j(\mathbf{o})$ is the probability of generating the observation vector $\mathbf{o}$ given that the HMM is in state j of the model.

The HMM, $\mathcal{M}$, may be written as

$$\mathcal{M} = \{\{a_{ij}\}, \{b_j(\cdot)\}\}$$

Training HMMs

Efficient training (and recognition) algorithms exist for HMMs. The standard training approach used is **Maximum Likelihood** (ML) estimation.



To train an HMM it is necessary to align each speech frame with a state (or Gaussian), gather statistics on the aligned data, and then update the model parameters.

The alignment may be done in a probabilistic fashion **Baum-Welch** re-estimation, or a hard *max* fashion **Viterbi** training. Baum-Welch re-estimation is an example of the

Expectation-Maximisation (EM) algorithm.

Expectation Maximisation

We have already seen an example of EM when training Gaussian mixture models. Training of HMMs is similar to training GMMs.

- for GMMs the "assignment" to a component is only dependent on the position of the features vector in acoustic space.
- for HMMs the "assignment" to a state will also depend on *when* the feature vector occurs in time.

The training algorithm is designed to guarantee that for each new estimate, the model set is more likely to generate the training data than the previous model set estimate, unless a *local* optimum has been reached.

$$p(\mathbf{O}|\hat{\mathcal{M}}) \geq p(\mathbf{O}|\mathcal{M})$$

- $\mathcal{M}$ is the new model set
- $\hat{\mathcal{M}}$ is the new model set
- $\mathbf{O}$ is the training observation sequence

By repeating this process many times a *local* maximum will be reached.

Baum-Welch Estimation

Baum-Welch estimation (EM for training HMMs) has two distinct stages.

1. **Expectation**: calculate the *posterior* probability of a feature vector being generated by a particular state. This probability (soft alignment) will be denoted as $L_i(t)$

   $$L_i(t) \;=\; P(s(t) = i|\mathbf{O}, \mathcal{M})$$

   In words: the probability that the feature vector at time t was generated by state i, given the whole training sequence and the current set of model parameters.

   The combination of the *observed* data $\mathbf{O}$ and the *unobserved* data

   $$\{\{L_2(1), \ldots, L_{N-1}(1)\}, \ldots, \{L_2(T), \ldots, L_{N-1}(T)\}\}$$

   is called the *complete dataset*;

2. **Maximisation**: using the complete dataset obtain the maximum likelihood estimate of the model parameters. The exact form of the parameter estimation depends on the HMM being trained.

These two stages are repeatedly applied in training the HMMs

Initial Model Estimates

To perform Baum-Welch training, initial estimates of the models are required. Possible initialisation schemes are:

- **phone-level labels**: if the database has been labelled at the phone level then these may be used in the training. The vast majority of speech databases (TIMIT is an exception) do not have these labels.
- **flat start**: all the models are initialised with the same parameters. These are typically set as the global mean and variance of all the training data.
- **best previous models**: use the best set of models on a "similar" database as the initial estimates for the new database.

In practice flat-start or best previous models are used.

If the posterior probability of state occupation, $L_i(t)$, was found by explicitly calculating every path through the model it would not be practical to train HMMs. Fortunately an efficient algorithm exists: the **Forward-Backward** algorithm.

The Forward-Backward algorithm

Two new variables are defined, the *forward* “probability”, $\alpha_i(t)$ defined as

$$\alpha_i(t) = p(\mathbf{o}_1, \ldots, \mathbf{o}_t, s(t) = i|\mathcal{M})$$

and the *backward* “probability”, $\beta_i(t)$, defined as

$$\beta_i(t) = p(\mathbf{o}_{t+1}, \ldots, \mathbf{o}_T|s(t) = i, \mathcal{M})$$

From the above, the frame alignment is given by

$$L_i(t) = \frac{\alpha_i(t)\beta_i(t)}{p(\mathbf{O}|\mathcal{M})}$$

There are efficient recursive (through time) routines to obtain the forward and backward probabilities. For $\alpha_i(t)$,

$$\alpha_j(t) \;=\; \left[\sum_{i=2}^{N-1} \alpha_i(t-1)a_{ij}\right] b_j(\mathbf{o}_t)$$

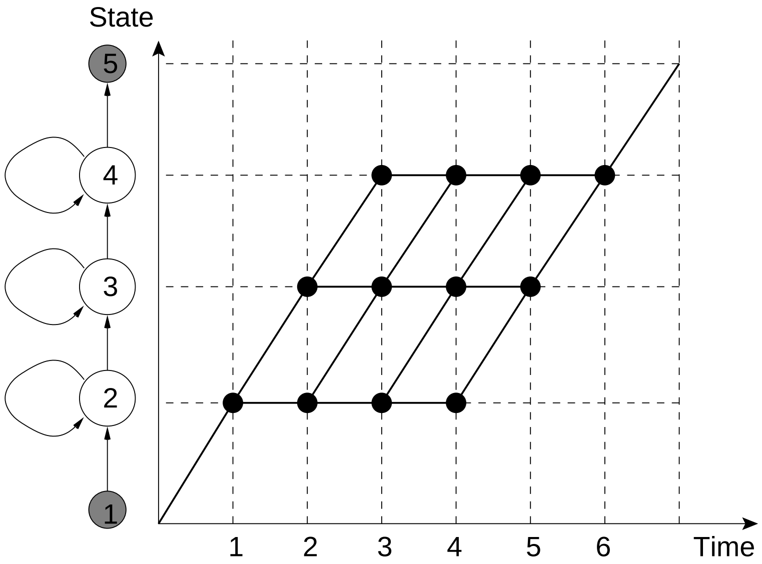
A similar expression exists for the backward probabilities

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij}b_j(\mathbf{o}_{t+1})\beta_j(t+1)$$

For the Viterbi case, the alignment involves finding only the most likely state sequence (using the Viterbi algorithm) and then assuming that (with probability 1) the model followed that state sequence.

Note: useful equality

$$p(\mathbf{O}|\mathcal{M}) = \sum_{i=2}^{N-1} \alpha_i(t)\beta_i(t) = \sum_{i=2}^{N-1} \alpha_i(T)a_{iN}$$



The elements of the **transition matrix**, a_{ij}, are defined as

$$a_{ij} \;=\; P(s(t+1) = j | s(t) = i)$$

and are estimated as

$$\hat{a}_{ij} = \frac{\text{Estimated number of transitions from states } i \text{ to } j}{\text{Estimated number of transitions from state } i}$$

In terms of the forward and backward probabilities

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_i(t) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_j(t+1)}{\sum_{t=1}^{T-1} \alpha_i(t) \beta_i(t)}$$

The transition probabilities give the state duration probabilities. It is simple to show from the state transition probabilities that

$$d_i(\tau) = a_{ii}^{\tau-1}(1 - a_{ii})$$

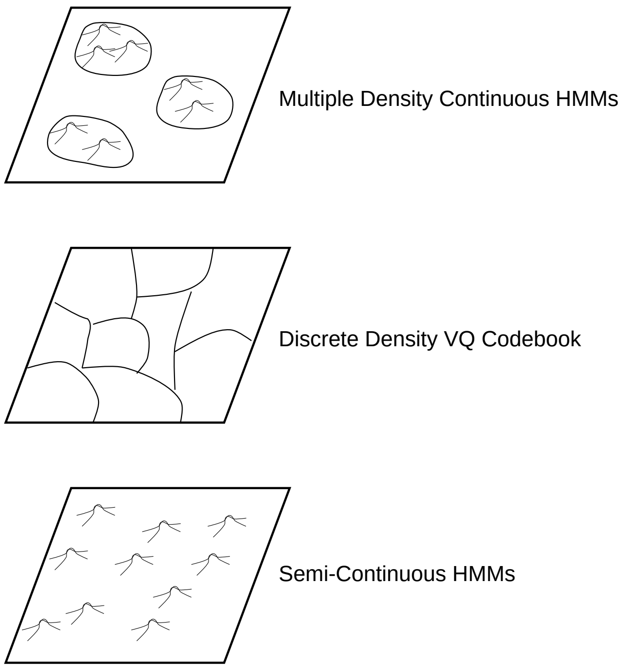
where $d_i(\tau)$ is the probability of staying in state i for τ frames.

Output Probability Distributions

HMMs are split into three broad classes according to the form of their output probability distributions (density functions).
The three classes are:

1. **Continuous Density HMMs**
2. **Discrete Density HMMs**
3. **Semi-Continuous (Tied-Mixture) HMMs**

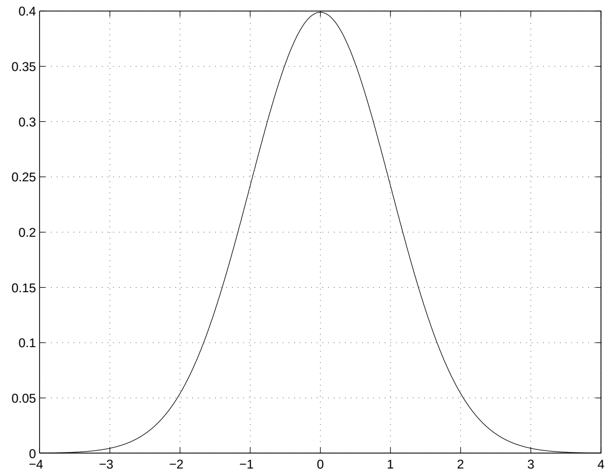
Each has a different representation of acoustic space.



The form of the output probability distribution alters the parameters of the HMMs and how they are estimated.

Continuous Density HMMs

In Speech Processing I we assumed that $b_j(\mathbf{o})$ is **Gaussian** distributed.



A single dimension Gaussian distribution is shown above: the parameters are the **mean** (μ) and **variance**, (σ^2). For an n-dimensional feature vector where all dimensions are uncorrelated (diagonal covariance matrix).

$$b_j(\mathbf{o}) = \prod_{d=1}^{n} \frac{1}{\sqrt{(2\pi\sigma_{jd}^2)}} \exp\left(\frac{-(o_d - \mu_{jd})^2}{2\sigma_{jd}^2}\right)$$

and for the full covariance case

$$b_j(\mathbf{o}) = \frac{1}{|\mathbf{\Sigma}_j|^{\frac{1}{2}}(2\pi)^{\frac{n}{2}}} \exp\left(-\frac{1}{2}(\mathbf{o} - \boldsymbol{\mu}_j)'\mathbf{\Sigma}_j^{-1}(\mathbf{o} - \boldsymbol{\mu}_j)\right)$$

where $\mathbf{\Sigma}_j$ is the *Covariance Matrix*.

It is common to write the probability density of a vector given a particular multivariate Gaussian distribution as

$$b_j(\mathbf{o}) = \mathcal{N}(\mathbf{o}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

Other distributions, (e.g. Laplacian) have also been used, however Gaussians are by far the most popular.

The parameters of a CDHMM are:

1. The transition matrix (standard HMM);
2. A mean and covariance for each state j

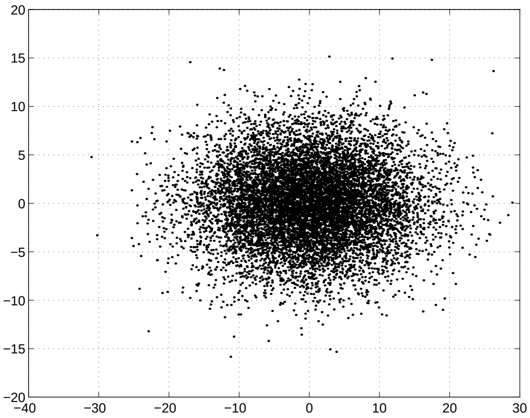
$$\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}$$

Number of Parameters

The number of parameters is very important when trying to ensure robust parameter estimation.

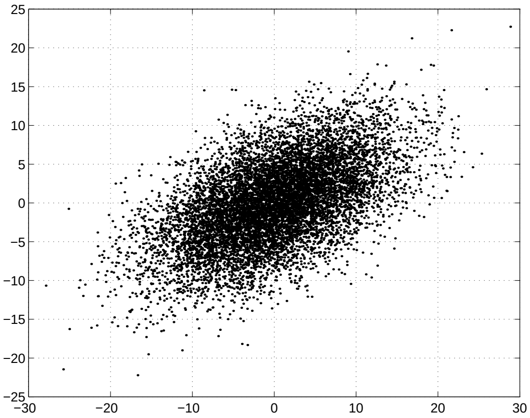
1. **Mean**: The number of parameters is the dimensionality of the data, n (39).
2. **Variances**: Two forms of covariance matrix are commonly used
   (a) **Diagonal**: Here the elements of the feature vector are assumed *uncorrelated*.
   $$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \sigma_n^2 \end{bmatrix}$$
   The number of parameters is again the dimensionality of the data, n (39).
   (b) **Full**: Elements may be correlated. A *symmetric* matrix is used. Number of parameters is $\frac{n(n+1)}{2}$ (780).
   $$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} \\ \sigma_{12} & \sigma_2^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1n} & \cdots & \cdots & \sigma_n^2 \end{bmatrix}$$

Uncorrelated Samples



$$\mathbf{\Sigma} = \begin{bmatrix} 64.00 & 0.00 \\ 0.00 & 16.00 \end{bmatrix}$$

Correlated Samples



$$\mathbf{\Sigma} = \begin{bmatrix} 52.00 & 21.00 \\ 21.00 & 28.00 \end{bmatrix}$$

Estimating CDHMM Parameters

The maximum likelihood estimates of the parameters of the output probability density of a CDHMM is straightforward once the frame/state alignment is known. The task is to obtain the mean and variance for a state.

- **Mean** vector

$$\hat{\boldsymbol{\mu}}_j = \frac{\text{Estimated sum of vectors emitted from state } j}{\text{Estimated number of vectors from state } j} = \frac{\sum_{t=1}^{T} L_j(t)\mathbf{o}_t}{\sum_{t=1}^{T} L_j(t)}$$

- **Full** covariance matrix

$$\hat{\boldsymbol{\Sigma}}_j = \frac{\text{Estimated sum of } (\mathbf{o}_t - \hat{\boldsymbol{\mu}}_j)(\mathbf{o}_t - \hat{\boldsymbol{\mu}}_j)' \text{ for state } j}{\text{Estimated number of vectors from state } j} = \frac{\sum_{t=1}^{T} L_j(t)(\mathbf{o}_t - \hat{\boldsymbol{\mu}}_j)(\mathbf{o}_t - \hat{\boldsymbol{\mu}}_j)'}{\sum_{t=1}^{T} L_j(t)}$$

- **Diagonal** covariance matrix

$$\hat{\sigma}_{jk}^2 = \frac{\text{Estimated sum of } (o_{tk} - \hat{\mu}_{jk})^2 \text{ for state } j}{\text{Estimated number of vectors from state } j} = \frac{\sum_{t=1}^{T} L_j(t)(o_{tk} - \hat{\mu}_{jk})^2}{\sum_{t=1}^{T} L_j(t)}$$

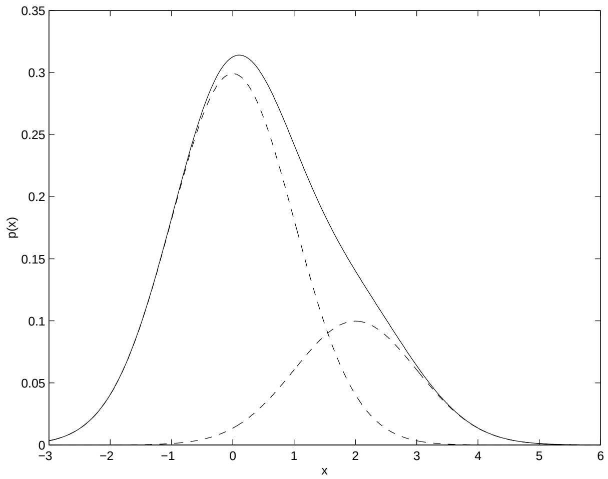
Mixture Distributions

Using a single (multivariate) Gaussian to model the output distribution may be poor. It is possible to use a mixture of Gaussians to model the distribution. Thus

$$b_j(\mathbf{o}) = \sum_{m=1}^{M} c_{jm} b_{jm}(\mathbf{o}) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(\mathbf{o}; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})$$

c_{jm} is the component weight, or prior. For this to be a probability density function it is necessary that

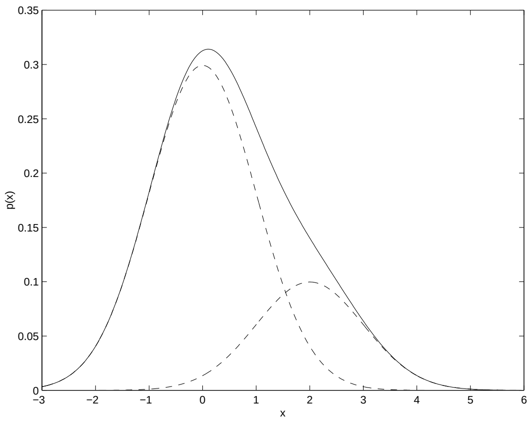
$$\sum_{m=1}^{M} c_{jm} = 1 \quad \text{and} \quad c_{jm} \geq 0$$



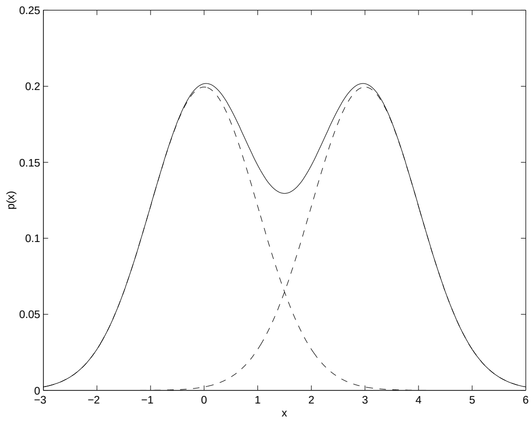
A two component Gaussian mixture distribution is shown above. Using Gaussian mixtures it is possible to approximate any distribution (provided you have enough components). Mixture distributions allow very flexible modelling of the acoustic vectors associated with a state.

Modelling PDFs

- **Asymmetric** and **Bimodal** distributions

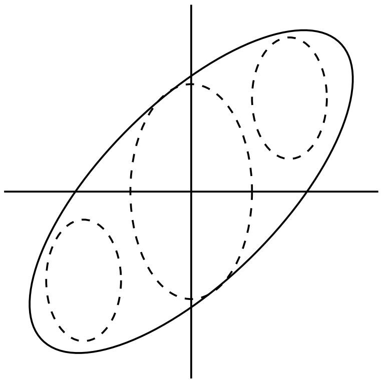


(a) Asymmetric Distribution

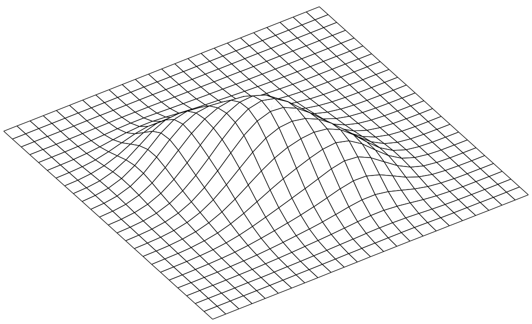


(b) Bimodal Distribution

- **Correlation-modelling** using *diagonal* covariance matrices



(c) Contours



(d) Distribution

Model Parameters

The parameters that need to be stored are

1. the transition matrix (standard HMM);
2. for *each* state the set of weights, means and variances

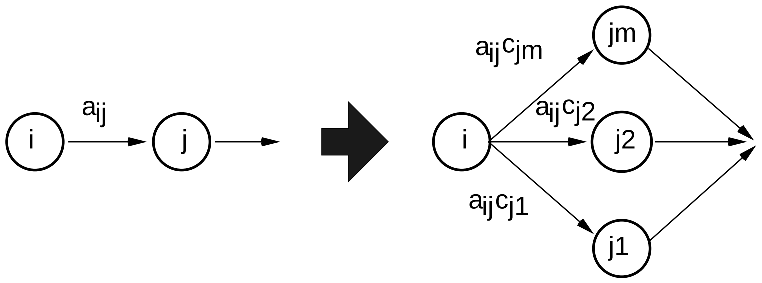
$$\{\{c_{j1}, \boldsymbol{\mu}_{j1}, \boldsymbol{\Sigma}_{j1}\}, \ldots, \{c_{jM}, \boldsymbol{\mu}_{jM}, \boldsymbol{\Sigma}_{jM}\}\}$$

Problem with the use of multiple component distributions is that it may result in a large number of Gaussians, hence system parameters.
Contrast parameters ($n = 39, M = 10$):

- **Single Full Covariance Gaussian**: mean requires n parameters, covariance matrix $\frac{n(n+1)}{2}$ - 819 parameters.
- **Single Diagonal Covariance Gaussian**: mean requires n parameters, covariance matrix n - 78 parameters.
- **Multiple Diagonal Covariance Gaussian Components**: M components require Mn parameters for the mean Mn parameters for the diagonal variances and $M - 1$ for weights - 789 parameters.

May be overcome using *tying* in the same fashion as clustering for triphones.

For Baum-Welch re-estimation each of the component Gaussians may be considered as a separate state thus



The alignment for a frame is to a particular Gaussian component of a particular state. Thus

$$\begin{aligned} L_{jm}(t) &= P(s(t) = jm|\mathbf{O}, \mathcal{M}) \\ &= \frac{1}{p(\mathbf{O}|\mathcal{M})} \sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij} c_{jm} b_{jm}(\mathbf{o}_t) \beta_j(t) \end{aligned}$$

The estimates of the mean and variance will be the same as for the single Gaussian case

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{t=1}^{T} L_{jm}(t)\mathbf{o}_t}{\sum_{t=1}^{T} L_{jm}(t)}$$

In addition, it is necessary to estimate the mixture weights. In a similar way to the transition probabilities

$$\hat{c}_{jm} = \frac{\text{Estimated Number of vectors from comp. } m \text{ state } j}{\text{Estimated number of vectors from state } j}$$

In terms of the alignments this becomes

$$\hat{c}_{jm} = \frac{\sum_{t=1}^{T} L_{jm}(t)}{\sum_{t=1}^{T} L_j(t)}$$

Training Gaussian Mixture HMMs

It is, again, necessary to have initial estimates of the models. If using *best previous set* no problem. However, not possible to use flat start. Two options:

1. **Clustering**. Gather together all aligned vectors for a state and used clustering (or VQ) techniques to initialise a set of Gaussians.
2. **Mixing-Up**. Iterative routine.