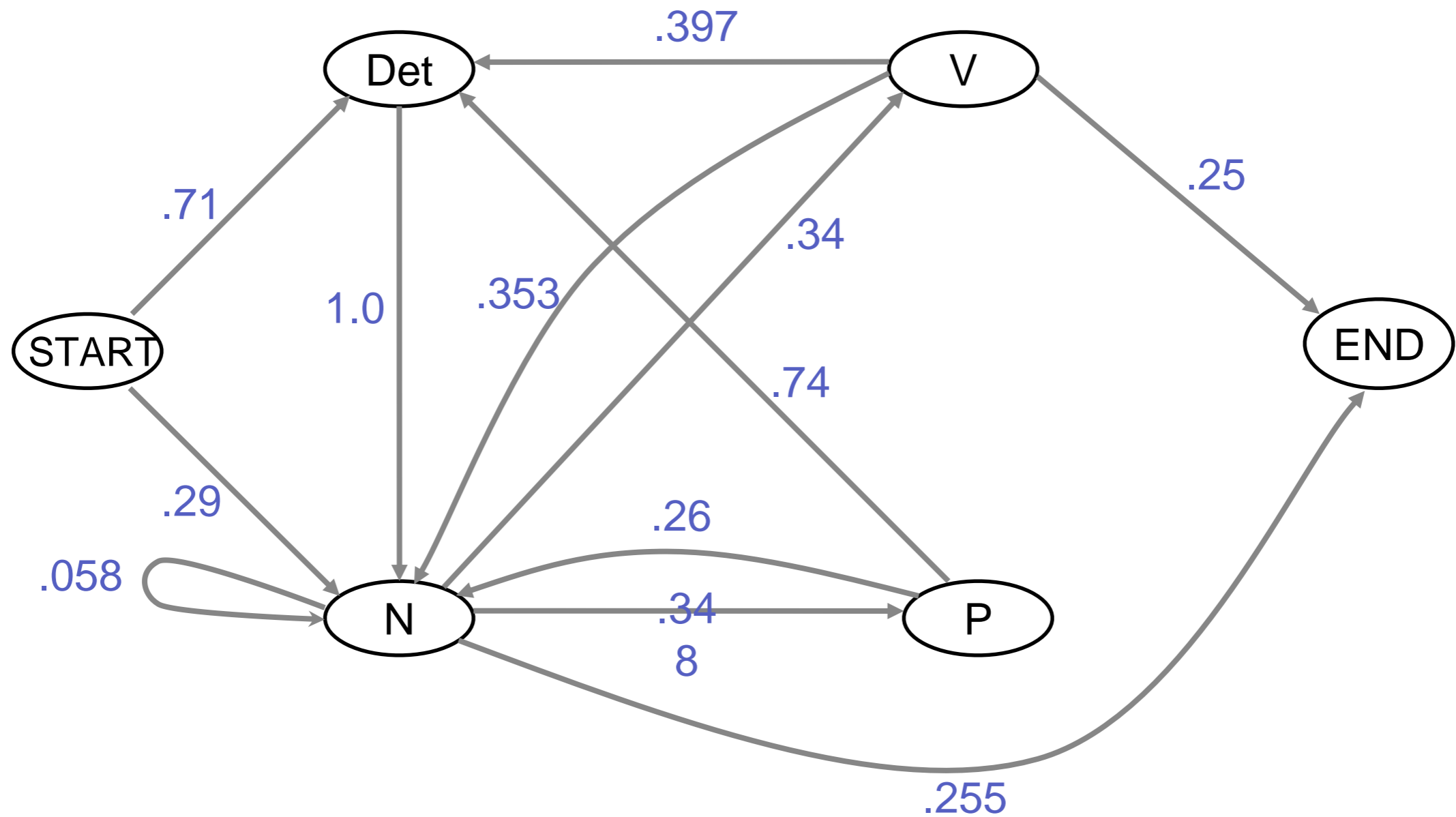


Hidden Markov Models

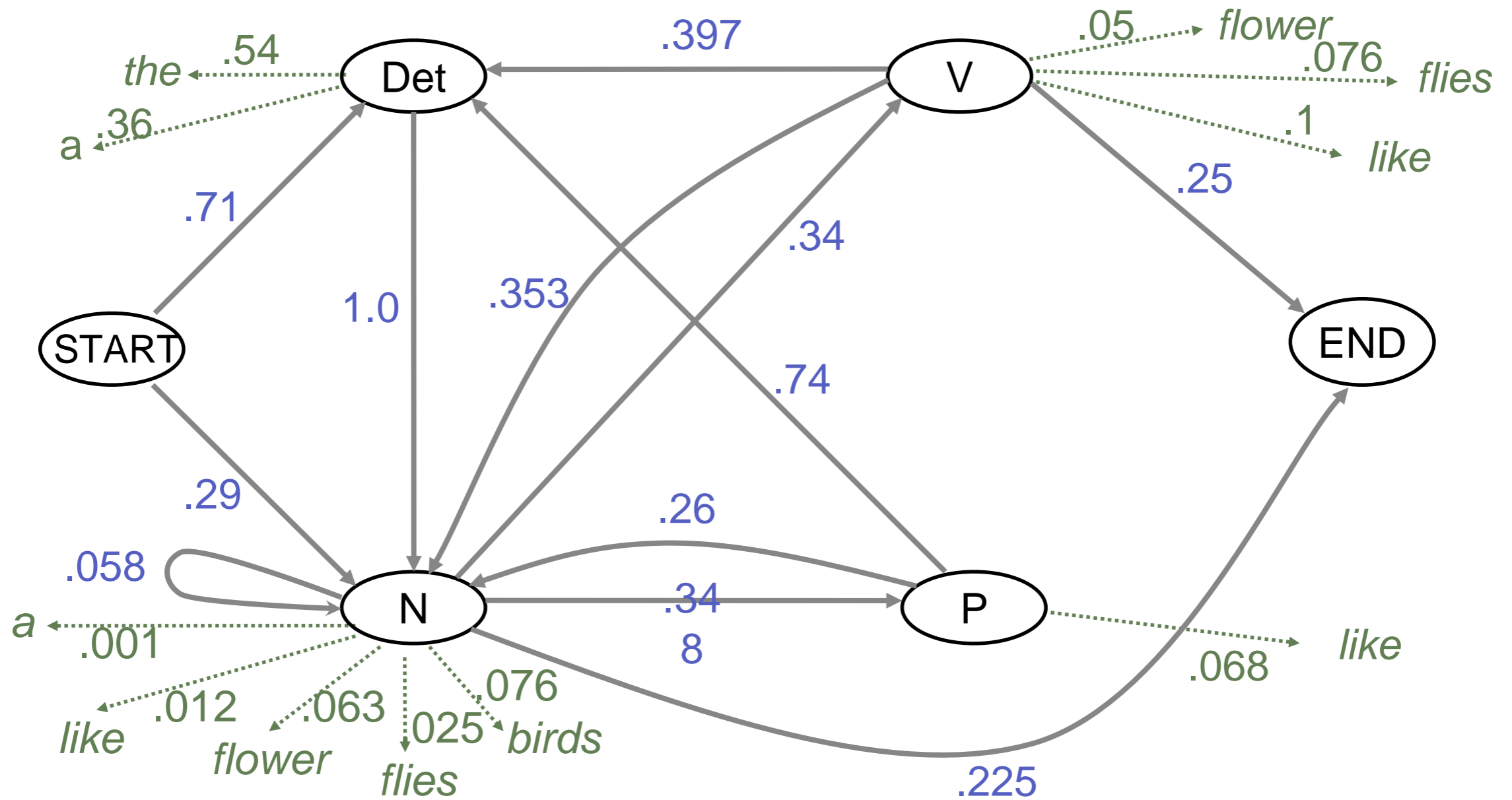
CSC401/2511 – Natural Language Computing –Spring 2026

Lecture 12 Gerald Penn (with slides from Steve Renals)

Markov model: Transition probabilities



HMM: MM with emission probabilities



$P(\text{the} \text{Det}) = .54$	$P(\text{like} \text{N}) = .012$	$P(\text{flower} \text{N}) = .063$	$P(\text{birds} \text{N}) = .076$
$P(\text{a} \text{Det}) = .36$	$P(\text{like} \text{V}) = .1$	$P(\text{flower} \text{V}) = .050$	$P(\text{flies} \text{V}) = .076$
$P(\text{a} \text{N}) = .001$	$P(\text{like} \text{P}) = .068$	$P(\text{flowers} \text{N}) = .050$	$P(\text{flies} \text{N}) = .025$
⋮	⋮	$P(\text{flowers} \text{V}) = .053$	⋮

Example HMM

Based on an example in section 7.3 of: Allen, James. *Natural Language Understanding* (2nd ed), 1995, Benjamin Cummings.

- Artificial corpus of PoS-tagged 300 sentences using only Det, N, V, P.

- *The flower flowers like a bird.*
Some birds like a flower with fruit beetles.
Like flies like flies.

...

- Some lexical generation probabilities:

$P(\textit{the} \text{Det}) = .54$	$P(\textit{like} \text{N}) = .012$	$P(\textit{flower} \text{N}) = .063$	$P(\textit{birds} \text{N}) = .076$
$P(\textit{a} \text{Det}) = .36$	$P(\textit{like} \text{V}) = .1$	$P(\textit{flower} \text{V}) = .050$	$P(\textit{flies} \text{V}) = .076$
$P(\textit{a} \text{N}) = .001$	$P(\textit{like} \text{P}) = .068$	$P(\textit{flowers} \text{N}) = .050$	$P(\textit{flies} \text{N}) = .025$
\vdots	\vdots	$P(\textit{flowers} \text{V}) = .053$	\vdots
		\vdots	

Markov model: Bigram table

Bigram C_{i-1}, C_i	Count C_{i-1}	Count C_{i-1}, C_i	$P(C_i C_{i-1})$	Estimate
START, Det	300	213	$P(\text{Det} \text{START})$	0.710
START, N	300	87	$P(\text{N} \text{START})$	0.290
Det, N	558	558	$P(\text{N} \text{Det})$	1.000
N, V	883	300	$P(\text{V} \text{N})$	0.340
N, N	883	51	$P(\text{N} \text{N})$	0.058
N, P	883	307	$P(\text{P} \text{N})$	0.348
N, END	883	225	$P(\text{END} \text{N})$	0.255
V, N	300	106	$P(\text{N} \text{V})$	0.353
V, Det	300	119	$P(\text{Det} \text{N})$	0.397
V, END	300	75	$P(\text{END} \text{V})$	0.250
P, Det	307	226	$P(\text{Det} \text{P})$	0.740
P, N	307	81	$P(\text{N} \text{P})$	0.260

Hidden Markov models

- Three basic questions of HMMs:
 - **Inference:** given a string of emissions, what is its probability?
 - **Decoding:** given a string of emissions, what is the most likely state sequence that produced it?
 - **Update:** given a (long) string of emissions, how can we improve the parameters of the model to make it more probable?

Hidden Markov models

- Given the observed output, we want to find the most likely path through the model.

The can will rust

det modal verb **modal verb** noun

noun noun **verb**

verb verb

Hidden Markov models

- At any state in an HMM, how you got there is irrelevant to computing the next transition.
 - So, just need to remember the best path and probability up to that point.
 - Define $\phi(c_{i-1})$ as the probability of the best sequence up to state C_{i-1} .
- Then find C_i that maximizes $\phi(C_{i-1}) \times P(C_i|C_{i-1}) \times P(w|C_i)$ **3** from slide 17

Viterbi Algorithm

- Given an HMM and an observation O of its output, finds the most probable sequence S of states that produced O .
 - O = words of sentence, S = PoS tags of sentence
- Parameters of HMM based on large training corpus.
- Then find C_i that maximizes $\varphi(C_{i-1}) \times P(C_i|C_{i-1}) \times P(w|C_i)$
 $\beta_i = C_{i-1}$ [*backtrace*]

Baum-Welch Algorithm

- Given an HMM M and an observation O , adjust the parameters of M to improve the probability $P(O)$.
 - O = words of sentence, $M = \langle \pi, A, B \rangle$
- This is an instance of Expectation-Maximization (EM).

HMMs and GMMs

- Key models and algorithms for HMM acoustic models
- Gaussians
- GMMs: Gaussian mixture models
- HMMs: Hidden Markov models
- HMM algorithms
 - Likelihood computation (forward algorithm)
 - Most probable state sequence (Viterbi algorithm)
 - Estimating the parameters (EM algorithm)

Fundamental Equation of Statistical Speech Recognition

If \mathbf{X} is the sequence of acoustic feature vectors (observations) and \mathbf{W} denotes a word sequence, the most likely word sequence \mathbf{W}^* is given by

$$\mathbf{W}^* = \arg \max_{\mathbf{W}} P(\mathbf{W} | \mathbf{X})$$

Applying Bayes' Theorem:

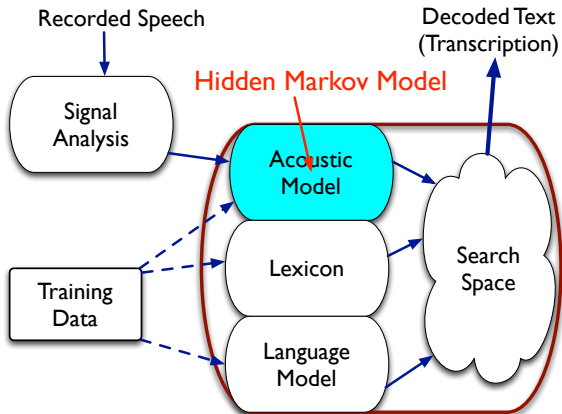
$$P(\mathbf{W} | \mathbf{X}) = \frac{p(\mathbf{X} | \mathbf{W}) P(\mathbf{W})}{p(\mathbf{X})}$$

$$\propto p(\mathbf{X} | \mathbf{W}) P(\mathbf{W})$$

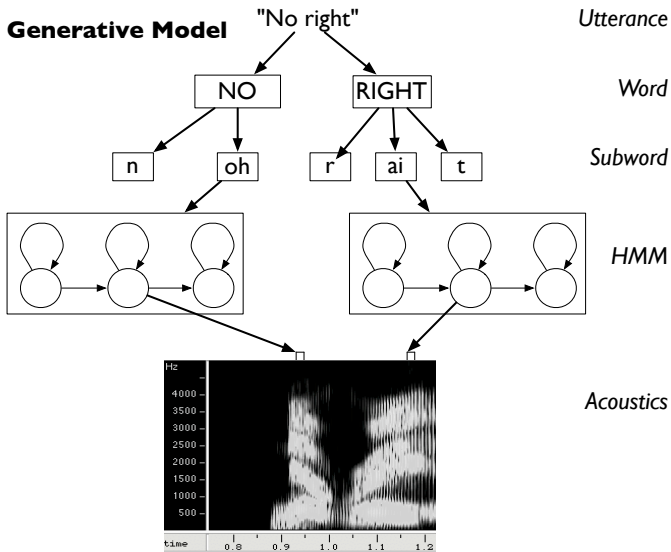
$$\mathbf{W}^* = \arg \max_{\mathbf{W}} \underbrace{p(\mathbf{X} | \mathbf{W})}_{\text{Acoustic model}} \underbrace{P(\mathbf{W})}_{\text{Language model}}$$

NB: \mathbf{X} is used hereafter to denote the output feature vectors from the signal analysis module rather than DFT spectrum.

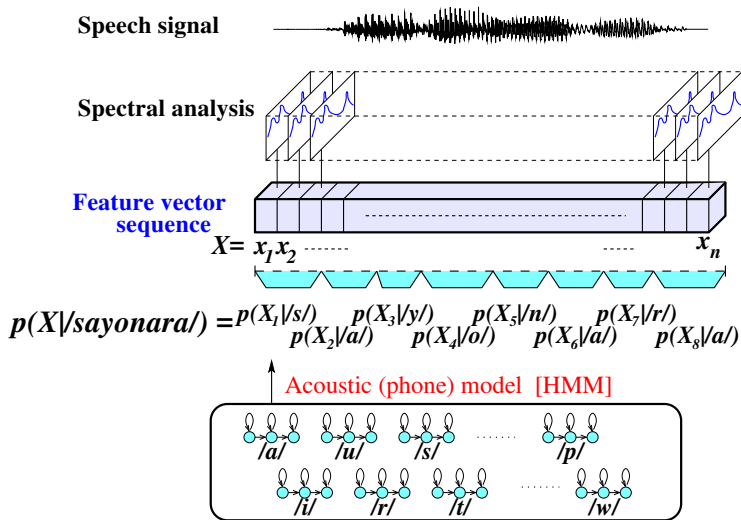
Acoustic Modelling



Hierarchical modelling of speech



Calculation of $p(X|W)$



NB: some conditional independency is assumed here.

How to calculate $p(X_1|/s/)$?

Assume $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{T_1}$ corresponds to phoneme $/s/$,
the *conditional probability* that we observe the sequence is

$$p(X_1|/s/) = p(\mathbf{x}_1, \dots, \mathbf{x}_{T_1}|/s/), \quad \mathbf{x}_i = (x_{1i}, \dots, x_{di})^t \in \mathcal{R}^d$$

We know that HMM can be employed to calculate this. (*Viterbi* algorithm, *Forward / Backward* algorithm)

To grasp the idea of probability calculation, let's consider an extremely simple case where the length of input sequence is just one ($T_1 = 1$), and the dimensionality of \mathbf{x} is one ($d = 1$), so that we don't need HMM.

$$p(X_1|/s/) \longrightarrow p(x_1|/s/)$$

How to calculate $p(X_1|s/)$? (cont.)

$p(x|s/)$: conditional probability
(conditional probability density function (*pdf*) of x)

- A *Gaussian / normal distribution* function could be employed for this:

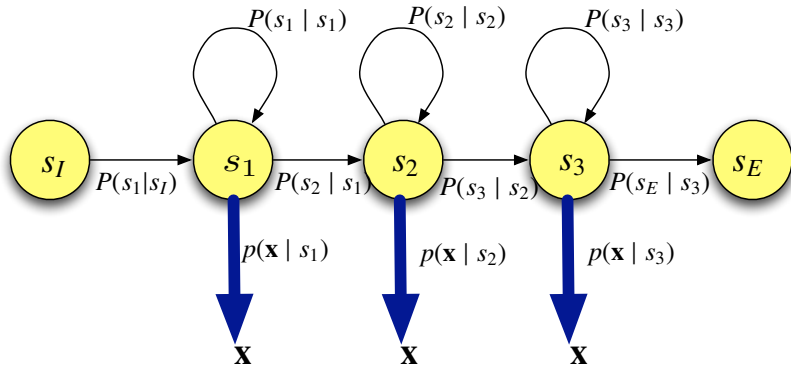
$$P(x|s/) = \frac{1}{\sqrt{2\pi\sigma_s^2}} e^{-\frac{(x-\mu_s)^2}{2\sigma_s^2}}$$

- The function has only two parameters, μ_s and σ_s^2
- Given a set of training samples $\{x_1, \dots, x_N\}$, we can estimate μ_s and σ_s

$$\hat{\mu}_s = \frac{1}{N} \sum_{i=1}^N x_i, \quad \hat{\sigma}_s^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu_s)^2$$

- For a general case where a phone lasts more than one frame, we need to employ HMM.

Acoustic Model: Continuous Density HMM



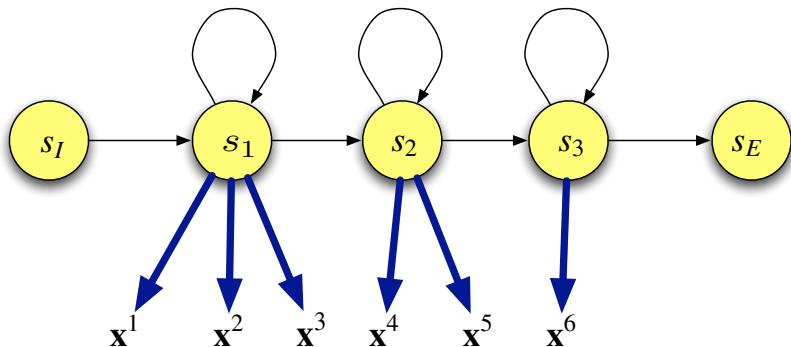
Probabilistic finite state automaton

Parameters λ :

- Transition probabilities: $a_{kj} = P(S=j | S=k)$
- Output probability density function: $b_j(\mathbf{x}) = p(\mathbf{x} | S=j)$

NB: Some textbooks use Q or q to denote the state variable S .
 \mathbf{x} corresponds to \mathbf{o}_t in Lecture slides 02.

Acoustic Model: Continuous Density HMM



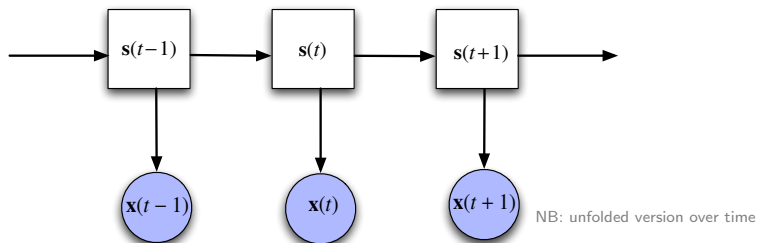
Probabilistic finite state automaton

Parameters λ :

- Transition probabilities: $a_{kj} = P(S=j | S=k)$
- Output probability density function: $b_j(\mathbf{x}) = p(\mathbf{x} | S=j)$

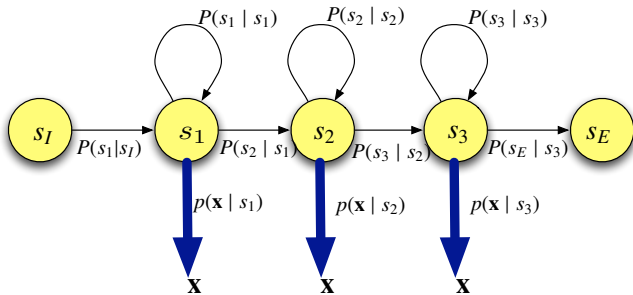
NB: Some textbooks use Q or q to denote the state variable S .
 \mathbf{x} corresponds to \mathbf{o}_t in Lecture slides 02.

HMM Assumptions



- 1 Markov process:** The probability of a state depends only on the previous state: $P(S(t) | S(t-1), S(t-2), \dots, S(1)) = P(S(t) | S(t-1))$
A state is conditionally independent of all other states given the previous state
- 2 Observation independence:** The output observation $\mathbf{x}(t)$ depends only on the state that produced the observation:
 $p(\mathbf{x}(t) | S(t), S(t-1), \dots, S(1), \mathbf{x}(t-1), \dots, \mathbf{x}(1)) = p(\mathbf{x}(t) | S(t))$
An acoustic observation \mathbf{x} is conditionally independent of all other observations given the state that generated it

Output distribution



- Single multivariate Gaussian with mean $\boldsymbol{\mu}_j$, covariance matrix $\boldsymbol{\Sigma}_j$:

$$b_j(\mathbf{x}) = p(\mathbf{x} | S=j) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

- M -component Gaussian mixture model:

$$b_j(\mathbf{x}) = p(\mathbf{x} | S=j) = \sum_{m=1}^M c_{jm} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})$$

- Neural network:

$$b_j(\mathbf{x}) \sim P(S=j | \mathbf{x}) / P(S=j) \quad \text{NB: NN outputs posterior probabilities}$$

Background: cdf

Consider a real valued random variable X

- Cumulative distribution function (cdf) $F(x)$ for X :

$$F(x) = P(X \leq x)$$

- To obtain the probability of falling in an interval we can do the following:

$$\begin{aligned}P(a < X \leq b) &= P(X \leq b) - P(X \leq a) \\ &= F(b) - F(a)\end{aligned}$$

- The rate of change of the cdf gives us the *probability density function* (pdf), $p(x)$:

$$p(x) = \frac{d}{dx}F(x) = F'(x)$$
$$F(x) = \int_{-\infty}^x p(x)dx$$

- $p(x)$ is **not** the probability that X has value x . But the pdf is proportional to the probability that X lies in a small interval centred on x .
- Notation: p for pdf, P for probability

The Gaussian distribution (univariate)

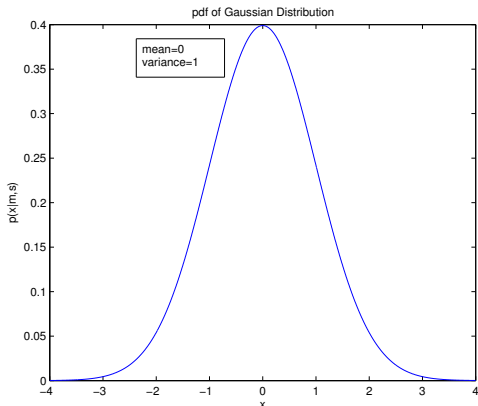
- The **Gaussian** (or **Normal**) distribution is the most common (and easily analysed) continuous distribution
- It is also a reasonable model in many situations (the famous “bell curve”)
- If a (scalar) variable has a Gaussian distribution, then it has a probability density function with this form:

$$p(x | \mu, \sigma^2) = \mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x - \mu)^2}{2\sigma^2}\right)$$

- The Gaussian is described by two parameters:
 - the mean μ (location)
 - the variance σ^2 (dispersion)

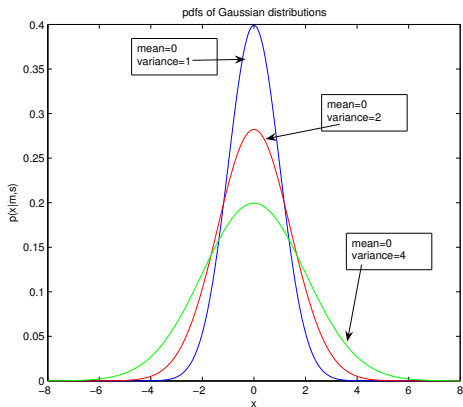
Plot of Gaussian distribution

- Gaussians have the same shape, with the location controlled by the mean, and the spread controlled by the variance
- One-dimensional Gaussian with zero mean and unit variance ($\mu = 0, \sigma^2 = 1$):



Properties of the Gaussian distribution

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x - \mu)^2}{2\sigma^2}\right)$$



Parameter estimation

- Estimate mean and variance parameters of a Gaussian from data x_1, x_2, \dots, x_T
- Use the following as the estimates:

$$\hat{\mu} = \frac{1}{T} \sum_{t=1}^T x_t \quad (\text{mean})$$

$$\hat{\sigma}^2 = \frac{1}{T} \sum_{t=1}^T (x_t - \hat{\mu})^2 \quad (\text{variance})$$

Exercise — maximum likelihood estimation (MLE)

Consider the log likelihood of a set of T training data points $\{x_1, \dots, x_T\}$ being generated by a Gaussian with mean μ and variance σ^2 :

$$\begin{aligned}L = \ln p(\{x_1, \dots, x_T\} | \mu, \sigma^2) &= -\frac{1}{2} \sum_{t=1}^T \left(\frac{(x_t - \mu)^2}{\sigma^2} - \ln \sigma^2 - \ln(2\pi) \right) \\ &= -\frac{1}{2\sigma^2} \sum_{t=1}^T (x_t - \mu)^2 - \frac{T}{2} \ln \sigma^2 - \frac{T}{2} \ln(2\pi)\end{aligned}$$

By maximising the the log likelihood function with respect to μ show that the maximum likelihood estimate for the mean is indeed the sample mean:

$$\mu_{ML} = \frac{1}{T} \sum_{t=1}^T x_t.$$

The multivariate Gaussian distribution

- The D -dimensional vector $\mathbf{x} = (x_1, \dots, x_D)^T$ follows a multivariate Gaussian (or normal) distribution if it has a probability density function of the following form:

$$p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

The pdf is parameterized by the mean vector $\boldsymbol{\mu} = (\mu_1, \dots, \mu_D)^T$

and the covariance matrix $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{11} & \dots & \sigma_{1D} \\ \vdots & \ddots & \vdots \\ \sigma_{D1} & \dots & \sigma_{DD} \end{pmatrix}$.

- The 1-dimensional Gaussian is a special case of this pdf
- The argument to the exponential $0.5(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})$ is referred to as a *quadratic form*.

Covariance matrix

- The mean vector $\boldsymbol{\mu}$ is the expectation of \mathbf{x} :

$$\boldsymbol{\mu} = E[\mathbf{x}]$$

- The covariance matrix $\boldsymbol{\Sigma}$ is the expectation of the deviation of \mathbf{x} from the mean:

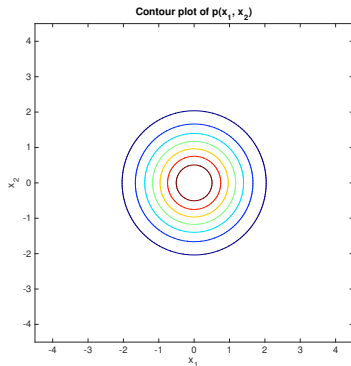
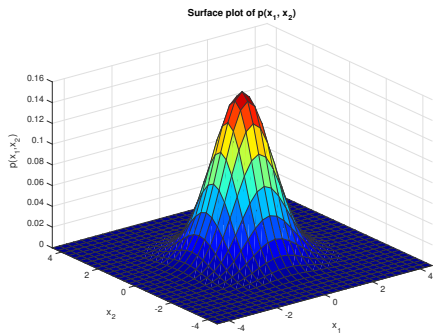
$$\boldsymbol{\Sigma} = E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T]$$

- $\boldsymbol{\Sigma}$ is a $D \times D$ symmetric matrix:

$$\sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)] = E[(x_j - \mu_j)(x_i - \mu_i)] = \sigma_{ji}$$

- The sign of the covariance helps to determine the relationship between two components:
 - If x_j is large when x_i is large, then $(x_i - \mu_i)(x_j - \mu_j)$ will tend to be positive;
 - If x_j is small when x_i is large, then $(x_i - \mu_i)(x_j - \mu_j)$ will tend to be negative.

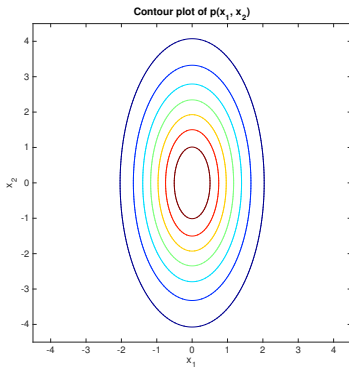
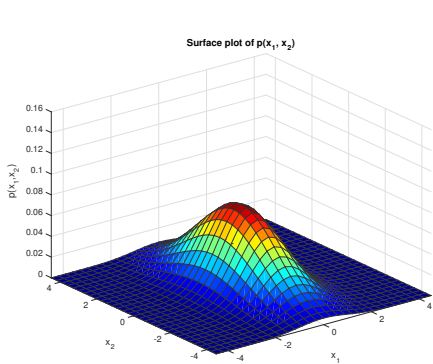
Spherical Gaussian



$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \rho_{12} = 0$$

NB: Correlation coefficient $\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$ ($-1 \leq \rho_{ij} \leq 1$)

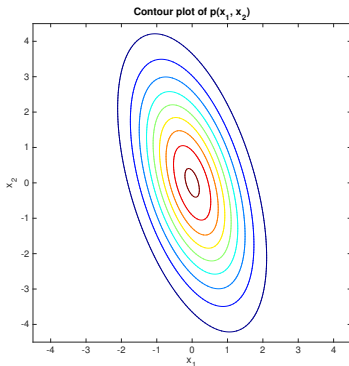
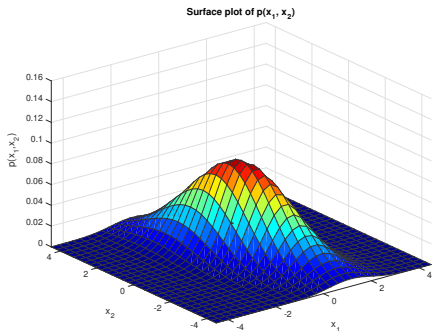
Diagonal Covariance Gaussian



$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix} \quad \rho_{12} = 0$$

NB: Correlation coefficient $\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$ ($-1 \leq \rho_{ij} \leq 1$)

Full covariance Gaussian



$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & -1 \\ -1 & 4 \end{pmatrix} \quad \rho_{12} = -0.5$$

NB: Correlation coefficient $\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$ ($-1 \leq \rho_{ij} \leq 1$)

Parameter estimation of a multivariate Gaussian distribution

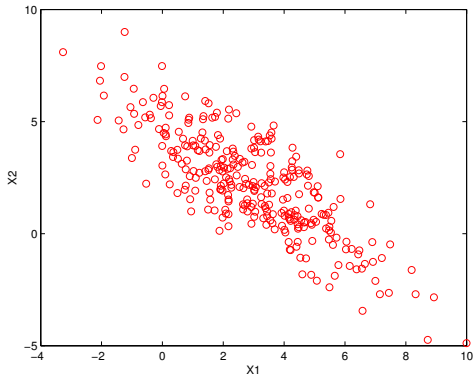
- It is possible to show that the mean vector $\hat{\boldsymbol{\mu}}$ and covariance matrix $\hat{\boldsymbol{\Sigma}}$ that maximize the likelihood of the training data are given by:

$$\hat{\boldsymbol{\mu}} = \frac{1}{T} \sum_{t=1}^T \mathbf{x}_t$$
$$\hat{\boldsymbol{\Sigma}} = \frac{1}{T} \sum_{t=1}^T (\mathbf{x}_t - \hat{\boldsymbol{\mu}})(\mathbf{x}_t - \hat{\boldsymbol{\mu}})^T$$

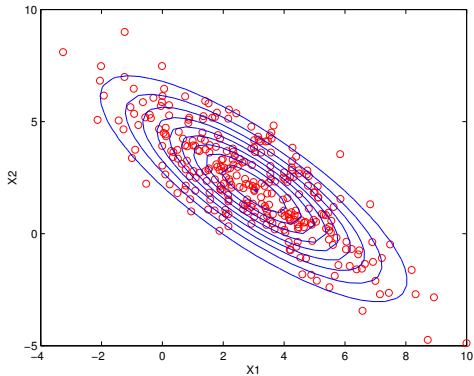
where $\mathbf{x}_t = (x_{t1}, \dots, x_{tD})^T$.

NB: T denotes either the number of samples or vector transpose depending on context.

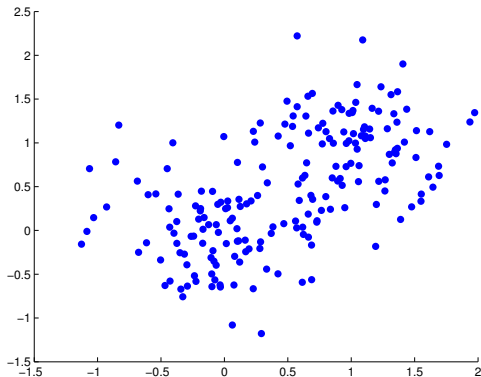
Example data



Maximum likelihood fit to a Gaussian



Data in clusters (example 1)



$$\boldsymbol{\mu}_1 = (0, 0)^T \quad \boldsymbol{\mu}_2 = (1, 1)^T \quad \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = 0.2 \mathbf{I}$$

Mixture model

- A more flexible form of density estimation is made up of a linear combination of component densities:

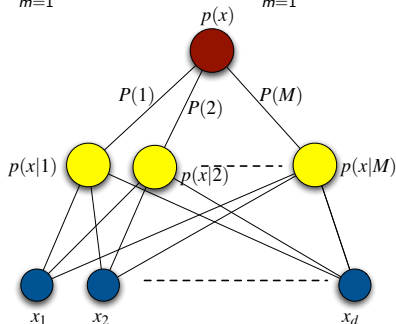
$$p(\mathbf{x}) = \sum_{m=1}^M p(\mathbf{x}|m)P(m)$$

- This is called a *mixture model* or a *mixture density*
- $p(\mathbf{x}|m)$: component densities
- $P(m)$: mixing parameters
- Generative model:
 - 1 Choose a mixture component based on $P(m)$
 - 2 Generate a data point \mathbf{x} from the chosen component using $p(\mathbf{x}|m)$

Gaussian mixture model

- The most important mixture model is the *Gaussian Mixture Model* (GMM), where the component densities are Gaussians
- Consider a GMM, where each component Gaussian $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$ has mean $\boldsymbol{\mu}_m$ and a **spherical covariance** $\boldsymbol{\Sigma}_m = \sigma_m^2 \mathbf{I}$

$$p(\mathbf{x}) = \sum_{m=1}^M P(m) p(\mathbf{x} | m) = \sum_{m=1}^M P(m) \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_m, \sigma_m^2 \mathbf{I})$$



GMM Parameter estimation when we know which component generated the data

- Define the indicator variable $z_{mt} = 1$ if component m generated data point \mathbf{x}_t (and 0 otherwise)
- If z_{mt} wasn't hidden then we could count the number of observed data points generated by m :

$$N_m = \sum_{t=1}^T z_{mt}$$

- And estimate the mean, variance and mixing parameters as:

$$\hat{\boldsymbol{\mu}}_m = \frac{\sum_t z_{mt} \mathbf{x}_t}{N_m}$$

$$\hat{\sigma}_m^2 = \frac{\sum_t z_{mt} \|\mathbf{x}_t - \hat{\boldsymbol{\mu}}_m\|^2}{N_m}$$

$$\hat{P}(m) = \frac{1}{T} \sum_t z_{mt} = \frac{N_m}{T}$$

GMM Parameter estimation when we don't know which component generated the data

- *Problem*: we don't know which mixture component a data point comes from...
- *Idea*: use the posterior probability $P(m|\mathbf{x})$, which gives the probability that component m was responsible for generating data point \mathbf{x} .

$$P(m|\mathbf{x}) = \frac{p(\mathbf{x}|m) P(m)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|m) P(m)}{\sum_{m'=1}^M p(\mathbf{x}|m') P(m')}$$

- The $P(m|\mathbf{x})$ s are called the *component occupation probabilities* (or sometimes called the *responsibilities*)
- Since they are posterior probabilities:

$$\sum_{m=1}^M P(m|\mathbf{x}) = 1$$

Soft assignment

- Estimate “soft counts” based on the component occupation probabilities $P(m | \mathbf{x}_t)$:

$$N_m^* = \sum_{t=1}^T P(m | \mathbf{x}_t)$$

- We can imagine assigning data points to component m weighted by the component occupation probability $P(m | \mathbf{x}_t)$
- So we could imagine estimating the mean, variance and prior probabilities as:

$$\hat{\boldsymbol{\mu}}_m = \frac{\sum_t P(m | \mathbf{x}_t) \mathbf{x}_t}{\sum_t P(m | \mathbf{x}_t)} = \frac{\sum_t P(m | \mathbf{x}_t) \mathbf{x}_t}{N_m^*}$$

$$\hat{\sigma}_m^2 = \frac{\sum_t P(m | \mathbf{x}_t) \|\mathbf{x}_t - \hat{\boldsymbol{\mu}}_m\|^2}{\sum_t P(m | \mathbf{x}_t)} = \frac{\sum_t P(m | \mathbf{x}_t) \|\mathbf{x}_t - \hat{\boldsymbol{\mu}}_m\|^2}{N_m^*}$$

$$\hat{P}(m) = \frac{1}{T} \sum_t P(m | \mathbf{x}_t) = \frac{N_m^*}{T}$$

EM algorithm

- *Problem!* Recall that:

$$P(m|\mathbf{x}) = \frac{p(\mathbf{x}|m)P(m)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|m)P(m)}{\sum_{m'=1}^M p(\mathbf{x}|m')P(m')}$$

We need to know $p(\mathbf{x}|m)$ and $P(m)$ to estimate the parameters of $P(m|\mathbf{x})$, and to estimate $P(m)$

- Solution: an iterative algorithm where each iteration has two parts:
 - Compute the component occupation probabilities $P(m|\mathbf{x})$ using the current estimates of the GMM parameters (means, variances, mixing parameters) (E-step)
 - Compute the GMM parameters using the current estimates of the component occupation probabilities (M-step)
- Starting from some initialization (e.g. using k-means for the means) these steps are alternated until convergence
- This is called the *EM Algorithm* and can be shown to maximize the likelihood

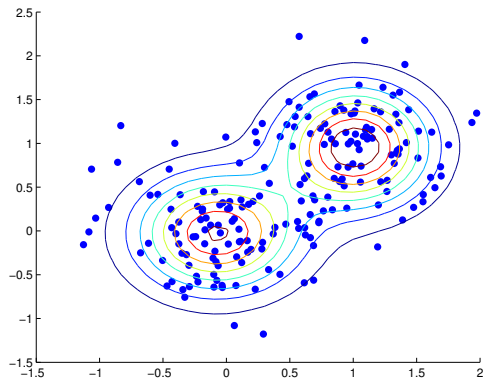
Maximum likelihood parameter estimation

- The likelihood of a data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T\}$ is given by:

$$\mathcal{L} = \prod_{t=1}^T p(\mathbf{x}_t) = \prod_{t=1}^T \sum_{m=1}^M p(\mathbf{x}_t | m) P(m)$$

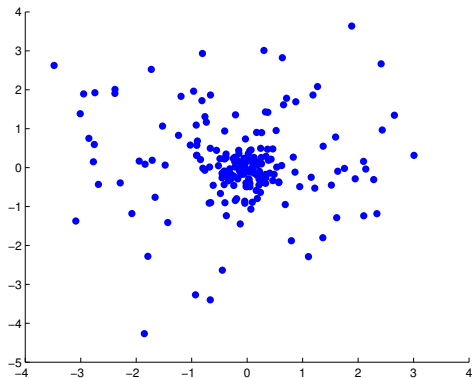
- We can regard the *negative log likelihood* as an error function:
- Considering the derivatives of E with respect to the parameters, gives expressions like the previous slide

Example 1 fit using a GMM



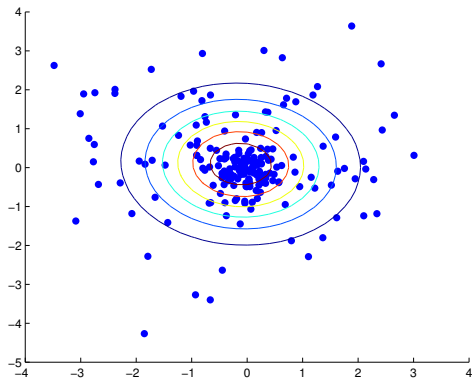
Fitted with a two component GMM using EM

Peakily distributed data (Example 2)



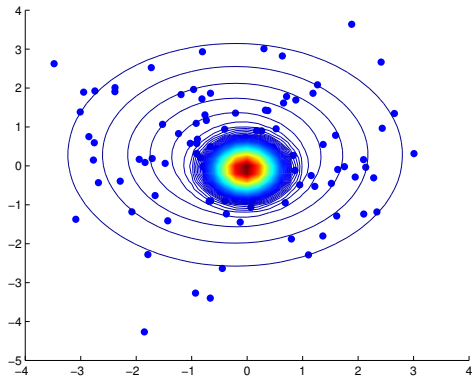
$$\mu_1 = \mu_2 = [0 \ 0]^T \quad \Sigma_1 = 0.1\mathbf{I} \quad \Sigma_2 = 2\mathbf{I}$$

Example 2 fit by a Gaussian



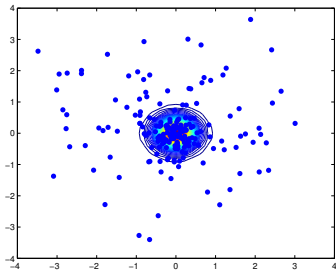
$$\mu_1 = \mu_2 = [0 \ 0]^T \quad \Sigma_1 = 0.1\mathbf{I} \quad \Sigma_2 = 2\mathbf{I}$$

Example 2 fit by a GMM

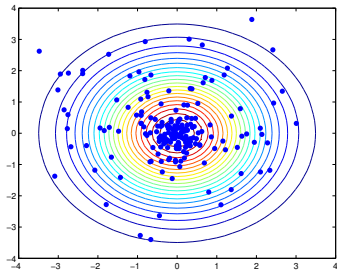


Fitted with a two component GMM using EM

Example 2: component Gaussians



$$P(\mathbf{x} | m=1)$$

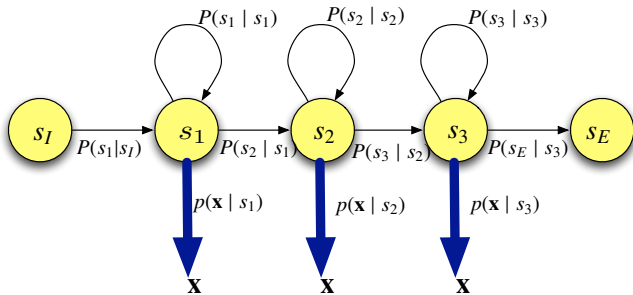


$$P(\mathbf{x} | m=2)$$

Comments on GMMs

- GMMs trained using the EM algorithm are able to self organize to fit a data set
- Individual components take responsibility for parts of the data set (probabilistically)
- Soft assignment to components not hard assignment — “soft clustering”
- GMMs scale very well, e.g.: large speech recognition systems can have 30,000 GMMs, each with 32 components: sometimes 1 million Gaussian components!! And the parameters all estimated from (a lot of) data by EM

Back to HMMs...



Output distribution:

- Single multivariate Gaussian with mean $\boldsymbol{\mu}_j$, covariance matrix $\boldsymbol{\Sigma}_j$:

$$b_j(\mathbf{x}) = p(\mathbf{x} | S=j) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

- M -component Gaussian mixture model:

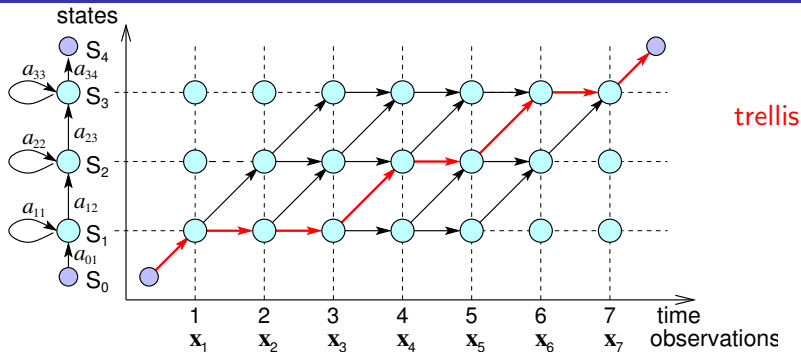
$$b_j(\mathbf{x}) = p(\mathbf{x} | S=j) = \sum_{m=1}^M c_{jm} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})$$

The three problems of HMMs

Working with HMMs requires the solution of three problems:

- ① **Likelihood** Determine the overall likelihood of an observation sequence $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_t, \dots, \mathbf{x}_T)$ being generated by an HMM.
- ② **Decoding** Given an observation sequence and an HMM, determine the most probable hidden state sequence
- ③ **Training** Given an observation sequence and an HMM, learn the best HMM parameters $\lambda = \{\{a_{jk}\}, \{b_j(\cdot)\}\}$

1. Likelihood: how to calculate?

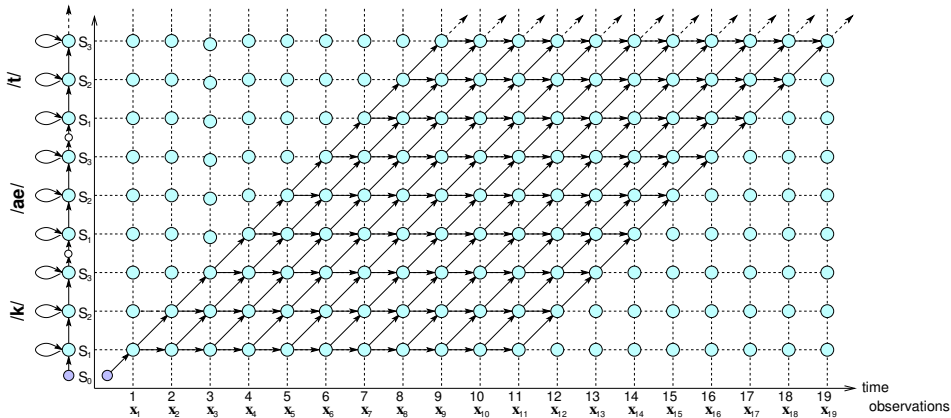


$$\begin{aligned} P(\mathbf{X}, \text{path}_\ell | \lambda) &= P(\mathbf{X} | \text{path}_\ell, \lambda) P(\text{path}_\ell | \lambda) \\ &= P(\mathbf{X} | s_0 s_1 s_1 s_1 s_2 s_2 s_3 s_3 s_4, \lambda) P(s_0 s_1 s_1 s_1 s_2 s_2 s_3 s_3 s_4 | \lambda) \\ &= b_1(x_1) b_1(x_2) b_1(x_3) b_2(x_4) b_2(x_5) b_3(x_6) b_3(x_7) a_{01} a_{11} a_{11} a_{12} a_{22} a_{23} a_{33} a_{34} \end{aligned}$$

$$P(\mathbf{X} | \lambda) = \sum_{\{\text{path}_\ell\}} P(\mathbf{X}, \text{path}_\ell | \lambda) \simeq \max_{\text{path}_\ell} P(\mathbf{X}, \text{path}_\ell | \lambda)$$

forward(backward) algorithm Viterbi algorithm

Trellis for /k ae t/



1. Likelihood: The Forward algorithm

- Goal: determine $p(\mathbf{X} | \lambda)$
- Sum over all possible state sequences $s_1 s_2 \dots s_T$ that could result in the observation sequence \mathbf{X}
- Rather than enumerating each sequence, compute the probabilities recursively (exploiting the Markov assumption)
- How many paths calculations in $p(\mathbf{X} | \lambda)$?

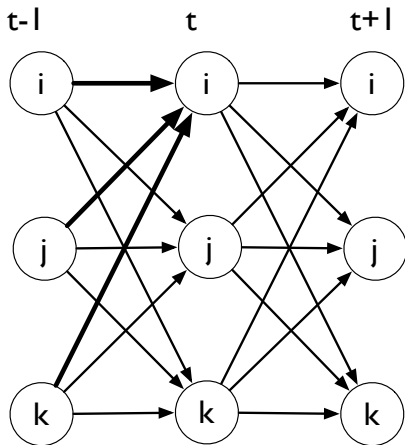
$$\sim \underbrace{N \times N \times \dots \times N}_{T \text{ times}} = N^T \quad \begin{array}{l} N : \text{ number of HMM states} \\ T : \text{ length of observation} \end{array}$$

e.g. $N^T \approx 10^{10}$ for $N=3$, $T=20$

- Computation complexity of multiplication: $O(2T N^T)$
- The Forward algorithm reduces this to $O(TN^2)$

Recursive algorithms on HMMs

Visualize the problem as a *state-time trellis*



1. Likelihood: The Forward algorithm

- Goal: determine $p(\mathbf{X} | \lambda)$
- Sum over all possible state sequences $s_1 s_2 \dots s_T$ that could result in the observation sequence \mathbf{X}
- Rather than enumerating each sequence, compute the probabilities recursively (exploiting the Markov assumption)
- *Forward probability*, $\alpha_t(j)$: the probability of observing the observation sequence $\mathbf{x}_1 \dots \mathbf{x}_t$ and being in state j at time t :

$$\alpha_t(j) = p(\mathbf{x}_1, \dots, \mathbf{x}_t, S(t)=j | \lambda)$$

1. Likelihood: The Forward recursion

- Initialization

$$\begin{aligned}\alpha_0(s_I) &= 1 \\ \alpha_0(j) &= 0 \quad \text{if } j \neq s_I\end{aligned}$$

- Recursion

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(\mathbf{x}_t) \quad 1 \leq j \leq N, 1 \leq t \leq T$$

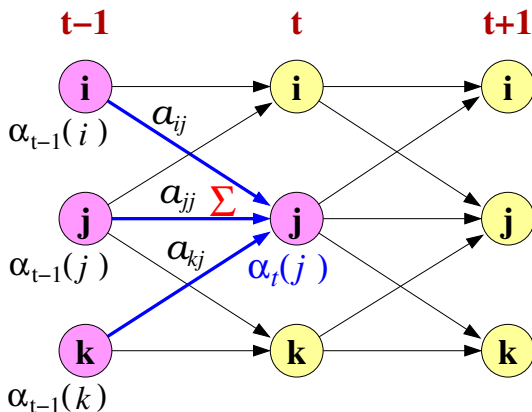
- Termination

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

s_I : initial state, s_E : final state

1. Likelihood: Forward Recursion

$$\alpha_t(j) = p(\mathbf{x}_1, \dots, \mathbf{x}_t, S(t)=j | \lambda) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(\mathbf{x}_t)$$



Viterbi approximation

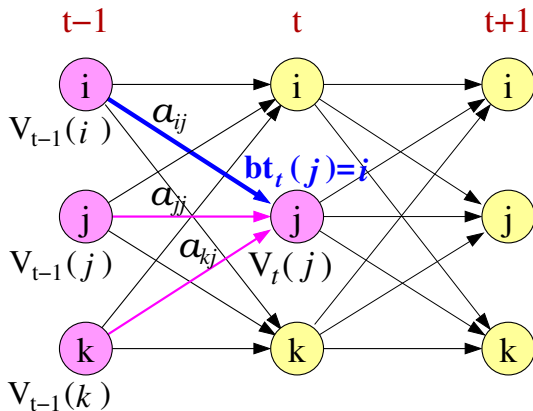
- Instead of summing over all possible state sequences, just consider the most likely
- Achieve this by changing the summation to a maximisation in the recursion:

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

- Changing the recursion in this way gives the likelihood of the most probable path
- We need to keep track of the states that make up this path by keeping a sequence of *backpointers* to enable a Viterbi *backtrace*: the backpointer for each state at each time indicates the previous state on the most probable path

Viterbi Recursion

Backpointers to the previous state on the most probable path



2. Decoding: The Viterbi algorithm

- Initialization

$$V_0(i) = 1$$

$$V_0(j) = 0 \quad \text{if } j \neq i$$

$$bt_0(j) = 0$$

- Recursion

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

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

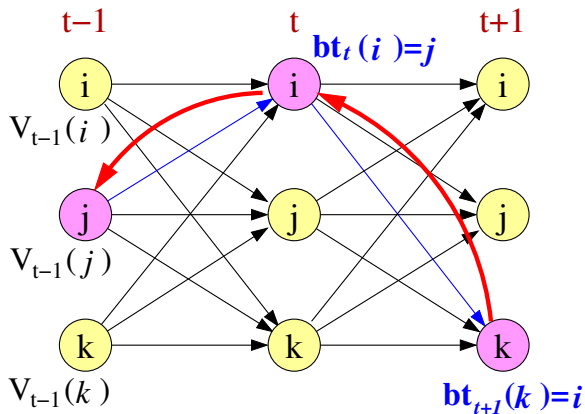
- Termination

$$P^* = V_T(s_E) = \max_{i=1}^N V_T(i) a_{iE}$$

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

Viterbi Backtrace

Backtrace to find the state sequence of the most probable path



3. Training: Forward-Backward algorithm

- Goal: Efficiently estimate the parameters of an HMM λ from an observation sequence
- Assume single Gaussian output probability distribution

$$b_j(\mathbf{x}) = p(\mathbf{x} | j) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

- Parameters λ :
 - Transition probabilities a_{ij} :

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

- Gaussian parameters for state j :
mean vector $\boldsymbol{\mu}_j$; covariance matrix $\boldsymbol{\Sigma}_j$

Viterbi Training

- If we knew the state-time alignment, then each observation feature vector could be assigned to a specific state
- A state-time alignment can be obtained using the most probable path obtained by Viterbi decoding
- Maximum likelihood estimate of a_{ij} , if $C(i \rightarrow j)$ is the count of transitions from i to j

$$\hat{a}_{ij} = \frac{C(i \rightarrow j)}{\sum_k C(i \rightarrow k)}$$

- Likewise if Z_j is the set of observed acoustic feature vectors assigned to state j , we can use the standard maximum likelihood estimates for the mean and the covariance:

$$\hat{\mu}_j = \frac{\sum_{\mathbf{x} \in Z_j} \mathbf{x}}{|Z_j|}$$

$$\hat{\Sigma}_j = \frac{\sum_{\mathbf{x} \in Z_j} (\mathbf{x} - \hat{\mu}_j)(\mathbf{x} - \hat{\mu}_j)^T}{|Z_j|}$$

EM Algorithm

- Viterbi training is an approximation—we would like to consider *all* possible paths
- In this case rather than having a hard state-time alignment we estimate a probability
- *State occupation probability*: The probability $\gamma_t(j)$ of occupying state j at time t given the sequence of observations.

Compare with component occupation probability in a GMM

- We can use this for an iterative algorithm for HMM training: the EM algorithm (whose adaption to HMM is called '[Baum-Welch algorithm](#)')
- Each iteration has two steps:
 - E-step** estimate the state occupation probabilities (Expectation)
 - M-step** re-estimate the HMM parameters based on the estimated state occupation probabilities (Maximisation)

Backward probabilities

- To estimate the state occupation probabilities it is useful to define (recursively) another set of probabilities—the *Backward* probabilities

$$\beta_t(j) = p(\mathbf{x}_{t+1}, \dots, \mathbf{x}_T | S(t)=j, \boldsymbol{\lambda})$$

The probability of future observations given a the HMM is in state j at time t

- These can be recursively computed (going backwards in time)
 - Initialisation

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

- Recursion

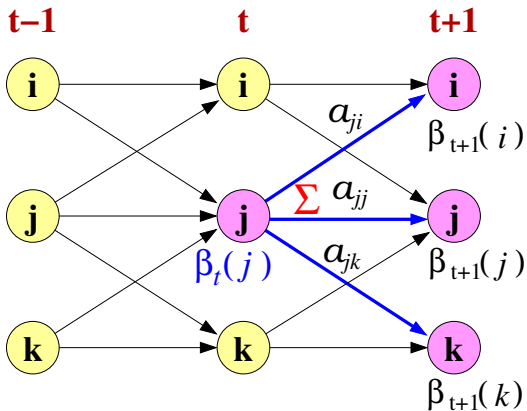
$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j) \quad \text{for } t = T-1, \dots, 1$$

- Termination

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

Backward Recursion

$$\beta_t(j) = p(\mathbf{x}_{t+1}, \dots, \mathbf{x}_T | S(t)=j, \lambda) = \sum_{j=1}^N a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(j)$$



State Occupation Probability

- The **state occupation probability** $\gamma_t(j)$ is the probability of occupying state j at time t given the sequence of observations
- Express in terms of the forward and backward probabilities:

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

recalling that $p(\mathbf{X} | \lambda) = \alpha_T(s_E)$

- Since

$$\begin{aligned} \alpha_t(j) \beta_t(j) &= p(\mathbf{x}_1, \dots, \mathbf{x}_t, S(t)=j | \lambda) \\ &\quad p(\mathbf{x}_{t+1}, \dots, \mathbf{x}_T | S(t)=j, \lambda) \\ &= p(\mathbf{x}_1, \dots, \mathbf{x}_t, \mathbf{x}_{t+1}, \dots, \mathbf{x}_T, S(t)=j | \lambda) \\ &= p(\mathbf{X}, S(t)=j | \lambda) \end{aligned}$$

$$P(S(t)=j | \mathbf{X}, \lambda) = \frac{p(\mathbf{X}, S(t)=j | \lambda)}{p(\mathbf{X} | \lambda)}$$

Re-estimation of Gaussian parameters

- The sum of state occupation probabilities through time for a state, may be regarded as a “soft” count
- We can use this “soft” alignment to re-estimate the HMM parameters:

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

Re-estimation of transition probabilities

- Similarly to the state occupation probability, we can estimate $\xi_t(i, j)$, the probability of being in i at time t and j at $t + 1$, given the observations:

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

- We can use this to re-estimate the transition probabilities

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

Pulling it all together

- 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 μ_j , covariance matrices Σ_j and transition probabilities a_{ij}
- The application of the EM algorithm to HMM training is sometimes called the Forward-Backward algorithm

Extension to a corpus of utterances

- We usually train from a large corpus of R utterances
- If \mathbf{x}_t^r is the t th frame of the r th utterance \mathbf{X}^r then we can compute the probabilities $\alpha_t^r(j)$, $\beta_t^r(j)$, $\gamma_t^r(j)$ and $\xi_t^r(i, j)$ as before
- The re-estimates are as before, except we must sum over the R utterances, eg:

$$\hat{\mu}_j = \frac{\sum_{r=1}^R \sum_{t=1}^T \gamma_t^r(j) \mathbf{x}_t^r}{\sum_{r=1}^R \sum_{t=1}^T \gamma_t^r(j)}$$

- In addition, we usually employ “*embedded training*”, in which fine tuning of phone labelling with “*forced Viterbi alignment*” or forced alignment is involved. (For details see Section 9.7 in Jurafsky and Martin’s SLP)

Extension to Gaussian mixture model (GMM)

- The assumption of a Gaussian distribution at each state is very strong; in practice the acoustic feature vectors associated with a state may be strongly non-Gaussian
- In this case an M -component Gaussian mixture model is an appropriate density function:

$$b_j(\mathbf{x}) = p(\mathbf{x} | S=j) = \sum_{m=1}^M c_{jm} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})$$

Given enough components, this family of functions can model any distribution.

- Train using the EM algorithm, in which the component estimation probabilities are estimated in the E-step

EM training of HMM/GMM

- Rather than estimating the state-time alignment, we estimate the component/state-time alignment, and component-state occupation probabilities $\gamma_t(j, m)$: the probability of occupying mixture component m of state j at time t .

($\xi_{tm}(j)$ in Jurafsky and Martin's SLP)

- We can thus re-estimate the mean of mixture component m of state j as follows

$$\hat{\mu}_{jm} = \frac{\sum_{t=1}^T \gamma_t(j, m) \mathbf{x}_t}{\sum_{t=1}^T \gamma_t(j, m)}$$

And likewise for the covariance matrices (mixture models often use diagonal covariance matrices)

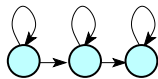
- The mixture coefficients are re-estimated in a similar way to transition probabilities:

$$\hat{c}_{jm} = \frac{\sum_{t=1}^T \gamma_t(j, m)}{\sum_{m'=1}^M \sum_{t=1}^T \gamma_t(j, m')}$$

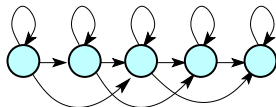
Doing the computation

- The forward, backward and Viterbi recursions result in a long sequence of probabilities being multiplied
- This can cause floating point *underflow* problems
- In practice computations are performed in the log domain (in which multiplies become adds)
- Working in the log domain also avoids needing to perform the exponentiation when computing Gaussians

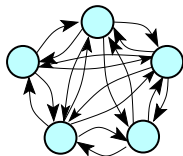
A note on HMM topology



left-to-right model



parallel path left-to-right model



ergodic model

$$\begin{pmatrix} a_{11} & a_{12} & 0 \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix}$$

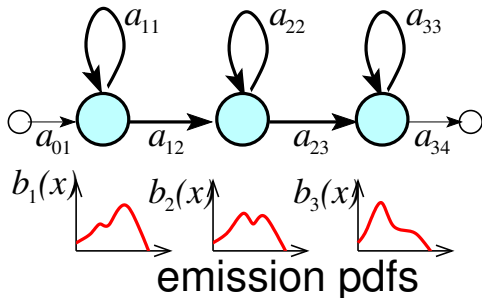
$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} \\ 0 & 0 & 0 & a_{44} & a_{45} \\ 0 & 0 & 0 & 0 & a_{55} \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix}$$

Speech recognition: left-to-right HMM with 3 ~ 5 states

Speaker recognition: ergodic HMM

A note on HMM emission probabilities



	Emission prob.	
Continuous (density) HMM	continuous density	GMM, NN/DNN
Discrete (probability) HMM	discrete probability	VQ
Semi-continuous HMM (tied-mixture HMM)	continuous density	tied mixture

Summary: HMMs

- HMMs provide a generative model for statistical speech recognition
- Three key problems
 - ① Computing the overall likelihood: the Forward algorithm
 - ② Decoding the most likely state sequence: the Viterbi algorithm
 - ③ Estimating the most likely parameters: the EM (Forward-Backward) algorithm
- Solutions to these problems are tractable due to the two key HMM assumptions
 - ① Conditional independence of observations given the current state
 - ② Markov assumption on the states

References: HMMs

- Gales and Young (2007). “The Application of Hidden Markov Models in Speech Recognition”, *Foundations and Trends in Signal Processing*, **1** (3), 195–304: section 2.2.
- Jurafsky and Martin (2008). *Speech and Language Processing* (2nd ed.): sections 6.1–6.5; 9.2; 9.4. (Errata at <http://www.cs.colorado.edu/~martin/SLP/Errata/SLP2-PIEV-Errata.html>)
- Rabiner and Juang (1989). “An introduction to hidden Markov models”, *IEEE ASSP Magazine*, **3** (1), 4–16.
- Renals and Hain (2010). “Speech Recognition”, *Computational Linguistics and Natural Language Processing Handbook*, Clark, Fox and Lappin (eds.), Blackwells.