Lecture 19

Input Processing

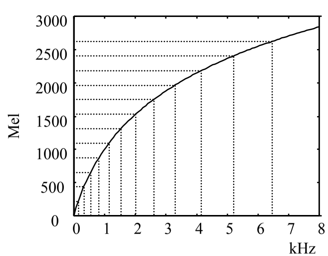
- Mel Frequency Scale
- Input Signal Preprocessing
  - Discrete Cosine Transform
  - Time Derivative estimates
  - Feature Decorrelation
  - Feature Vector length reduction
- Gaussian Mixtures
- Speech Recognition Summary

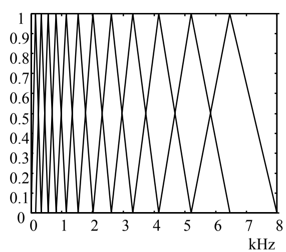
Feature Vector Requirements

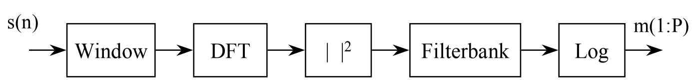
- When different people say the same phoneme, the feature vectors should have similar values.
- Different phonemes from the same or different speakers should give dissimilar values.
- For different examples of the same phoneme, the features should be independent and uncorrelated: this allows us to multiply their probabilities.
- For different examples of the same phoneme, each feature should preferably follow a probability distribution that is well described as a sum of gaussians.
- The features should not be affected by the amplitude of the speech signal otherwise recognition performance would vary with your distance from the microphone.

Mel Frequency Scale

- The feature vector must discriminate between speech sounds using as few components as possible to reduce computation.
- The human ear has better frequency resolution at low frequencies. The *mel* scale relates perceived pitch to frequency: linear at low f, logarithmic at high f:
  - $\text{mel}(f) = 2595 \log_{10}(1 + f / 700)$ where f is in Hz
  - Form a mel-spaced filterbank by setting the centre frequencies to equally spaced mel values.

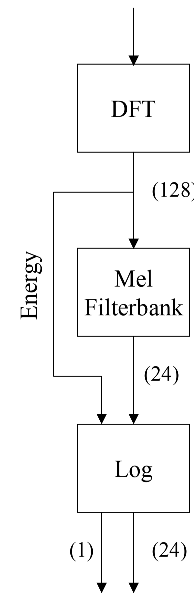






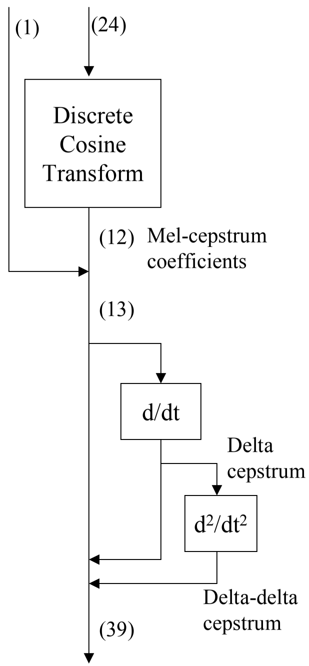
Preprocessing: Stage 1

- Divide signal into overlapping 25 ms segments at 10 ms intervals
- Apply Hamming window and take FFT
- Smooth the spectrum with a mel filterbank
  - mel filterbank concentrates data values in the more significant part of the spectrum
- Take the log of the mel spectrum
  - variations in signal level just cause a DC shift in the log spectrum
  - gaussian approximation is more nearly true for log spectrum than for the power spectrum directly



Preprocessing: Stage 2

- Discrete Cosine Transform (DCT)
  - reduces correlation between coefficients
  - compresses information into fewer low-order coefficients
  - output is the *mel-cepstrum*
  - DC component is ignored to make it independent of signal level
- First and Second time derivatives
  - provide additional information about how the spectrum is changing with time
- Result is a 39 element feature vector (or 38 if you drop the log energy).

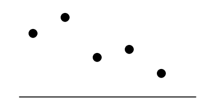


Discrete Cosine Transform

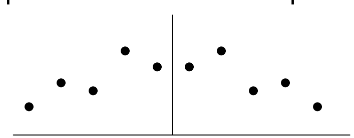
- The discrete cosine transform (DCT) of $m_1, \ldots, m_P$ is defined by

$$c_k = \sum_{p=1}^{P} m_p \cos\left(k(p-\tfrac{1}{2})\pi / P\right)$$

- The DCT of these points



is equal to the DFT of these points



with a phase shift to centre the time origin.

- Taking the DCT of the +ve frequency spectrum is essentially the same as taking the DFT of the symmetrical ±ve frequency spectrum.
- There are efficient algorithms for calculating the DCT

Polynomial Fitting

- To fit a polynomial to a set of points x_i, y_i

  for $i=1, 2, \ldots, N$: $y_i = \sum_{k=0}^{P} a_k x_i^k$

- Error $E = \sum_{i=1}^{N} e_i^2$ where $e_i = y_i - \sum_{k=0}^{P} a_k x_i^k$

- Minimize E by differentiating w.r.t. a_m, $m=0:P$

$$\frac{\partial E}{\partial a_m} = \sum_{i=1}^{N} 2e_i \frac{\partial e_i}{\partial a_m} = -2\sum_{i=1}^{N}\left(\left(y_i - \sum_{k=0}^{P} a_k x_i^k\right) \times x_i^m\right)$$

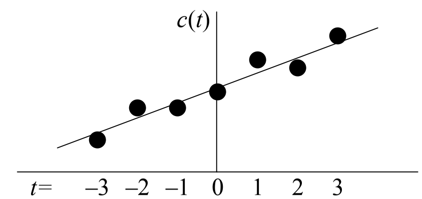
- Hence we get $P+1$ equations (same as LPC)

$$\sum_{k=0}^{P}\left(a_k \sum_{i=1}^{N} x_i^{k+m}\right) = \sum_{i=1}^{N} y_i x_i^m \quad for \quad m = 0,1,\ldots,P$$

- In matrix form (each value of m gives one row):

$$\begin{pmatrix} \sum x^0 & \sum x^1 & \sum x^2 & \cdots \\ \sum x^1 & \sum x^2 & \sum x^3 & \\ \sum x^2 & \sum x^3 & \sum x^4 & \\ \vdots & & & \ddots \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} \sum yx^0 \\ \sum yx^1 \\ \sum yx^2 \\ \vdots \end{pmatrix}$$

Cepstral Time-Derivatives



- Want to estimate dc/dt by fitting a line
  - Few points ⇒ noisy estimate
  - Many points ⇒ can't follow time variations
- Fit a 1st-order polynomial to $2T+1$ points:

$$\begin{pmatrix} \sum_{t=-T}^{T} t^0 & \sum_{t=-T}^{T} t^1 \\ \sum_{t=-T}^{T} t^1 & \sum_{t=-T}^{T} t^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{t=-T}^{T} c(t)t^0 \\ \sum_{t=-T}^{T} c(t)t^1 \end{pmatrix}$$

- this simplifies to

$$\begin{pmatrix} 2T+1 & 0 \\ 0 & \sum_{t=-T}^{T} t^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} \sum_{t=-T}^{T} c(t) \\ \sum_{t=-T}^{T} tc(t) \end{pmatrix} \Rightarrow a_1 = \frac{\sum_{t=-T}^{T} tc(t)}{\sum_{t=-T}^{T} t^2}$$

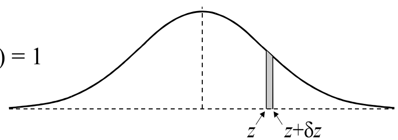
- Typically T=5 for 1st derivative and T=1 for 2nd

Multivariate Gaussian Distributions

- Z is a random variable with a standard Gaussian (or Normal) probability density func:

$$\text{pr}(Z \in [z, z+\delta z]) = (2\pi)^{-1/2} \exp\left(-\tfrac{1}{2}z^2\right)\delta z$$

- Mean: $\text{E}(Z) = 0$
  Variance: $\text{E}(Z^2) = 1$



- A linear sum of multiples of Gaussian random variables gives another Gaussian random variable. This property is unique to Gaussians.
- If we have a column random vector $\mathbf{Z}$ with P elements each of which is an *independent* standard Gaussian random variable then

$$\text{pr}(\mathbf{Z} \in [\mathbf{z}, \mathbf{z}+d\mathbf{z}]) = \prod_{i=1}^{P} (2\pi)^{-1/2} \exp\left(-\tfrac{1}{2}z_i^2\right) dz_i$$

$$= (2\pi)^{-P/2} \exp\left(-\tfrac{1}{2}\sum_{i=1}^{P} z_i^2\right) d\mathbf{z} = (2\pi)^{-P/2} \exp\left(-\tfrac{1}{2}\mathbf{z}^T\mathbf{z}\right) d\mathbf{z}$$

- Note too that because the z_i are independent.

$\text{E}(z_i z_j) = 0$ whenever $i \neq j \quad \Rightarrow \quad \text{E}(\mathbf{z}\,\mathbf{z}^T) = \mathbf{I}$

RECOG.PPT

Correlated Gaussian Distributions

- Now suppose that $\mathbf{x} = \mathbf{A}\,\mathbf{z}$ where $\mathbf{A}$ is an non-singular matrix, then $d\mathbf{x} = |\mathbf{A}|\, d\mathbf{z}$ and $\mathbf{z} = \mathbf{A}^{-1}\,\mathbf{x}$. Note that $\mathbf{X}$ is gaussian and $\text{E}(\mathbf{x})$ is 0.
- The covariance matrix of $\mathbf{x}$ is $\mathbf{C}=\text{E}(\mathbf{x}\mathbf{x}^T)$ and is symmetric and positive definite

$$\mathbf{C} = \text{E}\left(\mathbf{x}\mathbf{x}^T\right) = \text{E}\left(\mathbf{A}\mathbf{z}\mathbf{z}^T\mathbf{A}^T\right) = \mathbf{A}\text{E}(\mathbf{z}\mathbf{z}^T)\mathbf{A}^T = \mathbf{A}\mathbf{A}^T$$

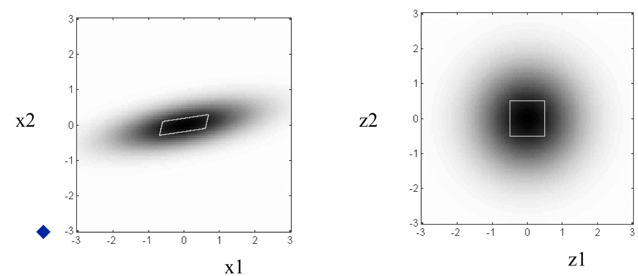
- We can work out the pdf of $\mathbf{x}$

$$\text{pr}(\mathbf{X} \in [\mathbf{x}, \mathbf{x}+d\mathbf{x}])\,/\,d\mathbf{x} = (2\pi)^{-P/2} \exp\left(-\tfrac{1}{2}\left(\mathbf{A}^{-1}\mathbf{x}\right)^T\left(\mathbf{A}^{-1}\mathbf{x}\right)\right)\,|\mathbf{A}|^{-1}$$

$$= (2\pi)^{-P/2}\,|\mathbf{A}|^{-1} \exp\left(-\tfrac{1}{2}\mathbf{x}^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{x}\right)$$

$$= (2\pi)^{-P/2}\,|\mathbf{C}|^{-1/2} \exp\left(-\tfrac{1}{2}\mathbf{x}^T\mathbf{C}^{-1}\mathbf{x}\right)$$

-



$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1.3 & 0.1 \\ 0.2 & 0.4 \end{pmatrix}\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \Rightarrow \mathbf{C} = \begin{pmatrix} 1.7 & 0.3 \\ 0.3 & 0.2 \end{pmatrix}$$ and $|\mathbf{C}| = |\mathbf{A}|^2 = 0.25$

Computational Costs

- The log prob density of correlated gaussians:

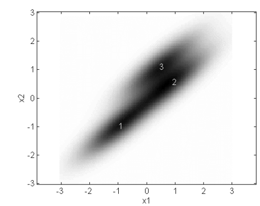
$$\log(\mathrm{pd}(x)) = \log\left((2\pi)^{-P/2}|\mathbf{C}|^{-1/2}\exp\left(-\tfrac{1}{2}\mathbf{x}^T\mathbf{C}^{-1}\mathbf{x}\right)\right)$$
$$= -\tfrac{1}{2}\left(P\log(2\pi) + \log(|C|) + \mathbf{x}^T\mathbf{C}^{-1}\mathbf{x}\right)$$

- The first two terms are independent of $\mathbf{x}$ and can be precalculated for each state.
- For F features, the final term involves F^2+F multiplications and F^2-1 additions: $39^2 = 1521$
- If the features are (or are assumed to be) independent, $\mathbf{C}$ is diagonal and we need $2F$ multiplications and $F-1$ additions
- Probability calculations consume most of the computation in a recogniser: almost all recognisers assume feature independence
  - DCT on log spectrum improves independence
  - We can do even better by applying a linear transformation to the feature vector.

Feature Decorrelation

- We can apply a linear transformation to our feature vectors, $\mathbf{x}$, to reduce correlations.

$\mathbf{x}$:



$\mathbf{W}_s$ is the covariance matrix of state s.

$\mathbf{W}$ is the average of the $\mathbf{W}_s$: the average within-state covariance matrix.

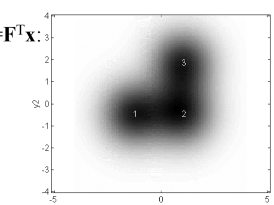
- If we multiply the feature vectors by a matrix $\mathbf{F}^T$, $\mathbf{y}=\mathbf{F}^T\mathbf{x}$, then the covariance matrix of $\mathbf{y}$ within state s is given by:

$$\mathrm{E}\left((\mathbf{y}-\bar{\mathbf{y}}_s)(\mathbf{y}-\bar{\mathbf{y}}_s)^T\right) = \mathrm{E}\left(\mathbf{F}^T(\mathbf{x}-\bar{\mathbf{x}}_s)(\mathbf{x}-\bar{\mathbf{x}}_s)^T\mathbf{F}\right) = \mathbf{F}^T\mathbf{W}_s\mathbf{F}$$

where $\bar{\mathbf{x}}_s$ is the mean value of $\mathbf{x}$ in state s.

- We transform our data with an $\mathbf{F}^T$ satisfying $\mathbf{F}^T\mathbf{W}\mathbf{F} = \mathbf{I}$.

$\mathbf{y}=\mathbf{F}^T\mathbf{x}$:



Eigenvectors

- d is an eigenvalue of $\mathbf{W}$ and $\mathbf{y}$ is an associated eigenvector if

$$\mathbf{Wy}=\mathbf{y}d$$

- Since $\mathbf{W}$ is symmetric and positive definite, we can find F orthonormal eigenvectors and make them the columns of a matrix:

$$\mathbf{WY}=\mathbf{YD}$$

  where $\mathbf{D}$ is a diagonal matrix of eigenvalues

- The orthonormality of the eigenvectors means that

$$\mathbf{Y}^T\mathbf{Y}=\mathbf{I}$$

- Now we define

$$\mathbf{F}=\mathbf{YD}^{-½}$$

- This gives

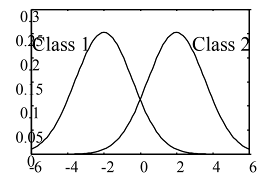
$$\mathbf{F}^T\mathbf{WF}=\mathbf{D}^{-½}\mathbf{Y}^T\mathbf{WYD}^{-½}=\mathbf{D}^{-½}\mathbf{Y}^T\mathbf{YDD}^{-½}=\mathbf{I}$$

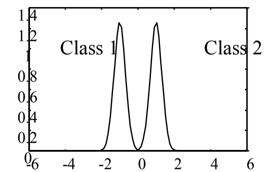
- In MATLAB:

```
[Y,D] = eig[W];
F = Y * sqrt(inv(D));
```


Class Discrimination

- We would like to make our feature vector a short as possible while preserving its ability to discriminate.





- The graphs show two possible distributions of a parameter for two different speech sounds (or *classes*).

- For a single parameter, Fisher's F Ratio is a measure of discriminability (the bigger the better): $F = \frac{\text{Variance of the class means}}{\text{Average variance within a class}}$

- For a parameter vector, this generalises to:

$$F = \text{trace}\left(\mathbf{W}^{-1}\mathbf{B}\right)$$

  where $\mathbf{W}$ and $\mathbf{B}$ are "average within-class" and "between-class" covariance matrices

RECOG.PPT 15/04/2002 8.83

Dimensionality Reduction

- We define **B** to be the between-state covariance matrix:

$$\mathbf{B} = \frac{1}{S}\sum_{s=1}^{S}\left(\bar{\mathbf{y}}_s - \bar{\bar{\mathbf{y}}}\right)\left(\bar{\mathbf{y}}_s - \bar{\bar{\mathbf{y}}}\right)^T$$

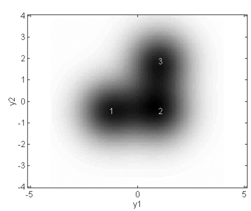
- As before we can find the eigenvalues of B

$$\mathbf{BG=GL}$$

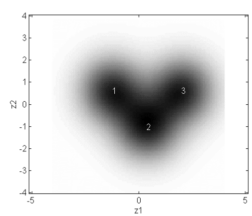
- where **G** is orthogonal and **L** diagonal.
- Set $\mathbf{z=G}^T\mathbf{y=G}^T\mathbf{F}^T\mathbf{x}$
- The between-state covariance matrix is now

$$\mathbf{G}^T\mathbf{BG=L}$$

- We can discard any elements of **z** for which the corresponding element of **L** is very small. Gives reduced feature set with equal (or even better) discrimination.



$\mathbf{y=F}^T\mathbf{x}$:



$\mathbf{z=G}^T\mathbf{y} = \mathbf{G}^T\mathbf{F}^T\mathbf{x}$:

RECOG.PPT

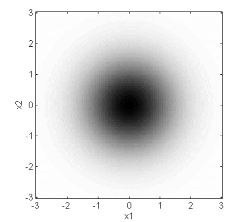
Gaussian Mixtures

- For large vocabularies, independent gaussian model is too simple. Use instead a mixture of gaussians:

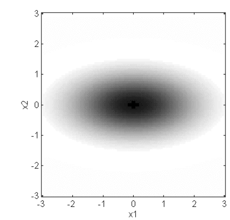
$$\text{pd}(\mathbf{x}) = \sum_{i=1}^{K} w_i N(\mathbf{m}_i, \mathbf{C}_i) \quad \text{with} \quad \sum_{i=1}^{K} w_i = 1$$

$$\text{where} \quad N(\mathbf{m}_i, \mathbf{C}_i) = (2\pi)^{-\frac{1}{2}P} \,|\mathbf{C}|^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}\mathbf{x}^T\mathbf{C}^{-1}\mathbf{x}\right)$$

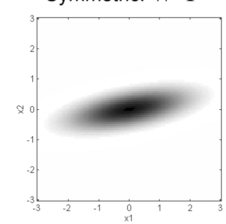
For simplicity we restrict **C** to be diagonal.



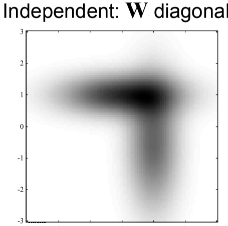
Symmetric: **W=I**



Independent: **W** diagonal

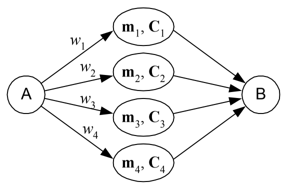


Correlated



Diagonal Gaussian Mixture

Mixtures = Alternate HMM states



- The total probability of all paths from A to B is the sum of the individual path probabilities

$$\mathrm{pd}(\mathbf{x}) = \sum_{i=1}^{K} w_i N(\mathbf{m}_i, \mathbf{C}_i)$$

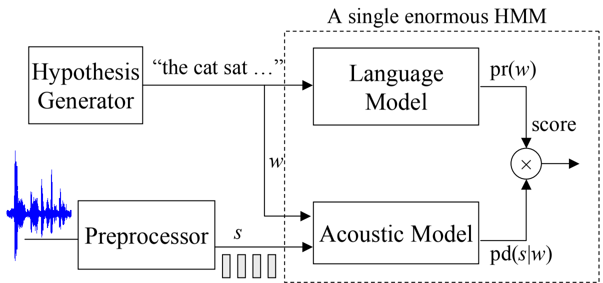
this is identical to the gaussian mixture expression.

- Once we have initial values for the model parameters we can use *Viterbi* and *Baum-Welch* procedures to train them.
- We can view gaussian mixtures as describing alternative pronunciations of a particular speech sound

K-means Algorithm

- We need to form an initial estimate for the K mixture means, $\mathbf{m}_i$, and covariances, $\mathbf{C}_i$.
- First create and train models with only one mixture using Viterbi training.
- Use Viterbi alignment to determine which training frames correspond to each state.
- For each state
  - Set the $\mathbf{m}_i$ to K randomly chosen training frames
  - Repeat until convergence occurs:
    - Allocate each training frame to whichever $\mathbf{m}_i$ it is nearest to.
    - Update each $\mathbf{m}_i$ to the mean of all the frames that were allocated to it
    - If no frames were allocated to $\mathbf{m}_i$, set it to a randomly chosen point from one of the other distributions.
  - Set $\mathbf{C}_i$ to the covariance of the frames allocated to $\mathbf{m}_i$.

Speech Recognition



- Preprocessor
  - Mel Cepstrum + Velocity + Acceleration
  - Linear Transform to decorrelate & reduce F
- Acoustic Model
  - 60,000 triphones × 3 states × 20 features × 10 mixtures = 72,000,000 parameters to train.
- Language Model
  - Phonetic description of each word in vocabulary + trigram or quadram transition probabilities
- Dynamic model creation
  - Create storage only for models when needed: use pruning to delete models with a hopelessly low probability.
  - Trade-off memory/computation versus accuracy