2. Isolated word speech recognition & HMMs

Goal: Recognition of isolated spoken words from a vocabulary

Problems:
- variable speed
- speaker independence
- prosody etc.

How do we hear?

→ cochlea, basilar membrane, inner hair cells, ... auditory cortex

1A. Signal pre-processing

- Sample the pressure-time function $f(t)$, digitise, highest freq. in speech signal ~ 10 KHz
  → sample with 20 KHz

- Frequency analysis: apply local Fourier transform with sliding window

$$c(\omega,t) = \int_{-\infty}^{\infty} w(t-t') f(t') e^{i\omega t'} dt'$$

simplest case: $w(t) = \begin{cases} 1 & \text{if } |t| \le b \\ 0 & \text{otherwise} \end{cases}$

choice of b: lowest freq. vs. time resolution

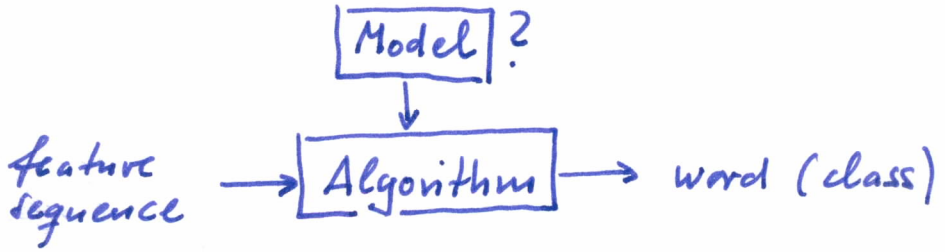
- Energy in spectra (logarithmic, dB)

$$S(\omega,t) = 20 \cdot \log_{10} \sqrt{Re^2 c(\omega,t) + Im^2 c(\omega,t)}$$

discretise domain of ν into ~20 frequency channels with freq. dependent width

- possibly also cluster spectral vectors
  - ⊕: small number of feature vectors
  - ⊖: dominance of stationary parts

B. Dynamic time warping & word recognition



Model: a set of prototypes ($\hat{=}$ feature sequences) for each word (class)

We need: distance measure for sequences of feature vectors:

prototype $\bar{x} = (\vec{x}_1, \ldots, \vec{x}_n)$, signal $\bar{y} = (\vec{y}_1, \ldots, \vec{y}_m)$

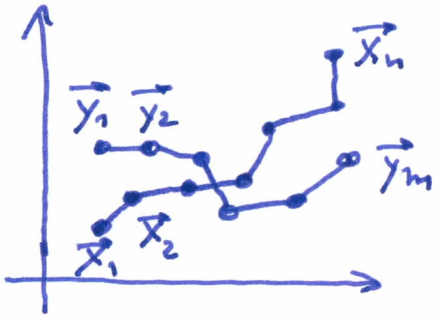
distance $D(\bar{x}, \bar{y}) = ?$

Monotonous matching (aka time warping)

$\tau = ((i_1, j_1), (i_2, j_2), \ldots, (i_c, j_c))$

$\tau \in T$ if

- $(i_1, j_1) = (1,1)$, $(i_c, j_c) = (n, m)$
- $i_{k-1} \leq i_k \leq i_{k-1} + 1$
- similar for j

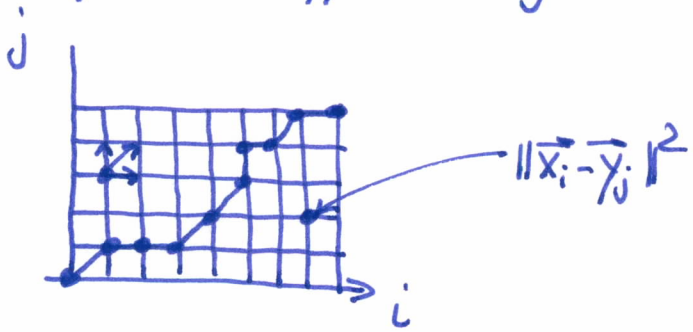


distance for a fixed matching $\tau \in \mathcal{T}$

$$D(\bar{x}, \bar{y}; \tau) = \sum_{k=1}^{l} \| \vec{x}_{i_k} - \vec{y}_{j_k} \|^2$$

distance $D(\bar{x}, \bar{y}) = \min_{\tau \in \mathcal{T}} D(\bar{x}, \bar{y}; \tau)$

How to compute it efficiently?



i.e. shortest path, here by dynamic programming, complexity $O(nm)$

Evaluate model & algorithm:

⊖ · many prototypes per word (class) → redundance, slows down inference

· learning: how to choose optimal prototypes?

Better: model each word (class) by an HMM

$X = (\vec{x}_1, \ldots, \vec{x}_n)$ - sequence of features

$S = (s_1, \ldots, s_n)$ - sequence of hidden states

$$P(X, S) = p(s_1) \prod_{i=2}^{n} p(s_i | s_{i-1}) \prod_{i=1}^{n} p(\vec{x}_i | s_i)$$

⊕ · fast inference

· feasible learning (of model parameters)