# Feature Extraction and Comparing Sequences

## DT2119 Speech and Speaker Recognition

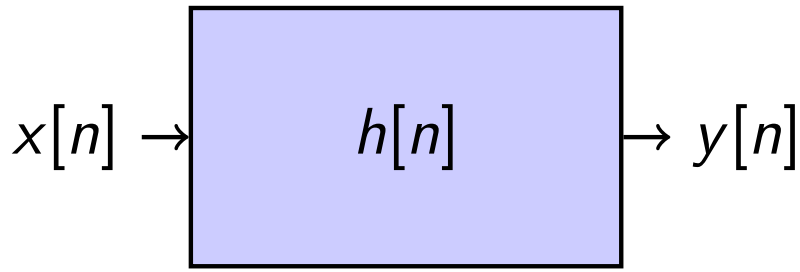
Giampiero Salvi

KTH/CSC/TMH giampi@kth.se

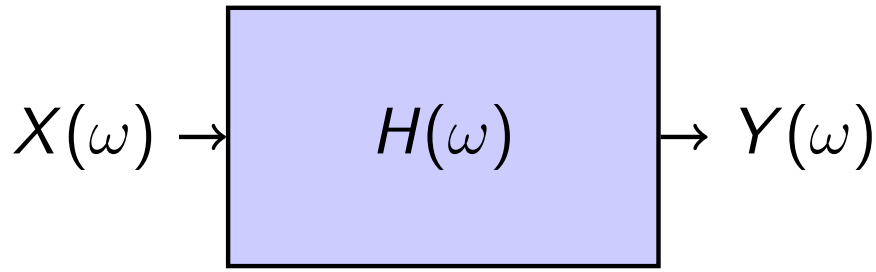
VT 2017

# Linear Time-Invariant Systems

Time Domain




Frequency Domain




Convolution

$$\begin{aligned} y[n] &= x[n] * h[n] \\ &= \sum_{k=-\infty}^{\infty} x[k]h[n-k] \end{aligned}$$

Multiplication

$$Y(\omega) = X(\omega)H(\omega)$$

$h[n]$ Impulse Response

$H(\omega)$ Transfer Function

# Fourier Transforms

Fourier transform of continuous signals

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{j\omega t} dt$$

Fourier transform of discrete signals

$$X(\omega) = \sum_{k=-\infty}^{\infty} x[k] e^{j\omega k}$$

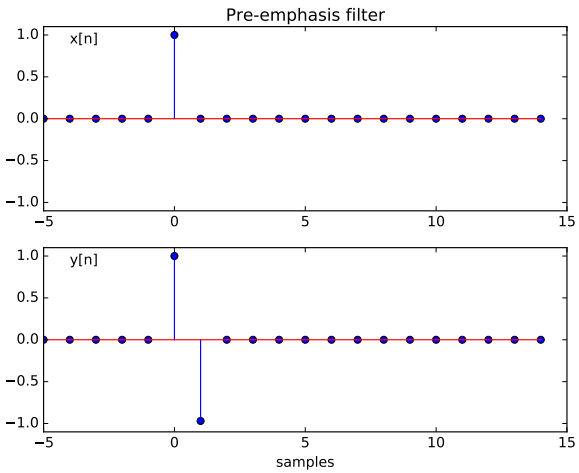
Discrete Fourier Transform (implemented as FFT)

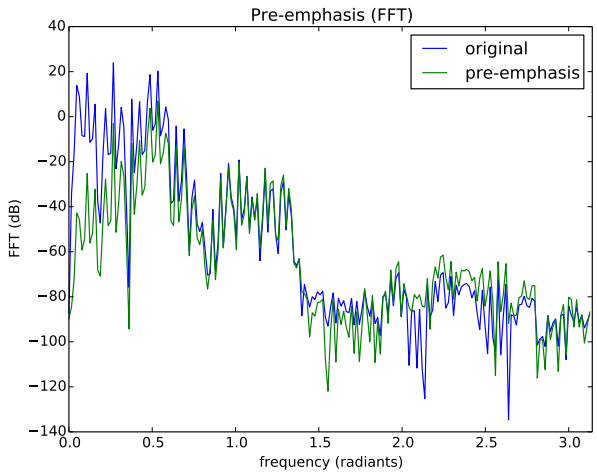
$$X[n] = \sum_{k=-\infty}^{\infty} x[k] e^{j2\pi \frac{n}{N} k}$$

# Properties of Fourier Transform

| Time domain | | Frequency domain |
|---|---|---|
| $x[n] * h[n]$ | $\Longleftrightarrow$ | $X(\omega)H(\omega)$ |

Example: any filter (pre-emphasis, moving average...)

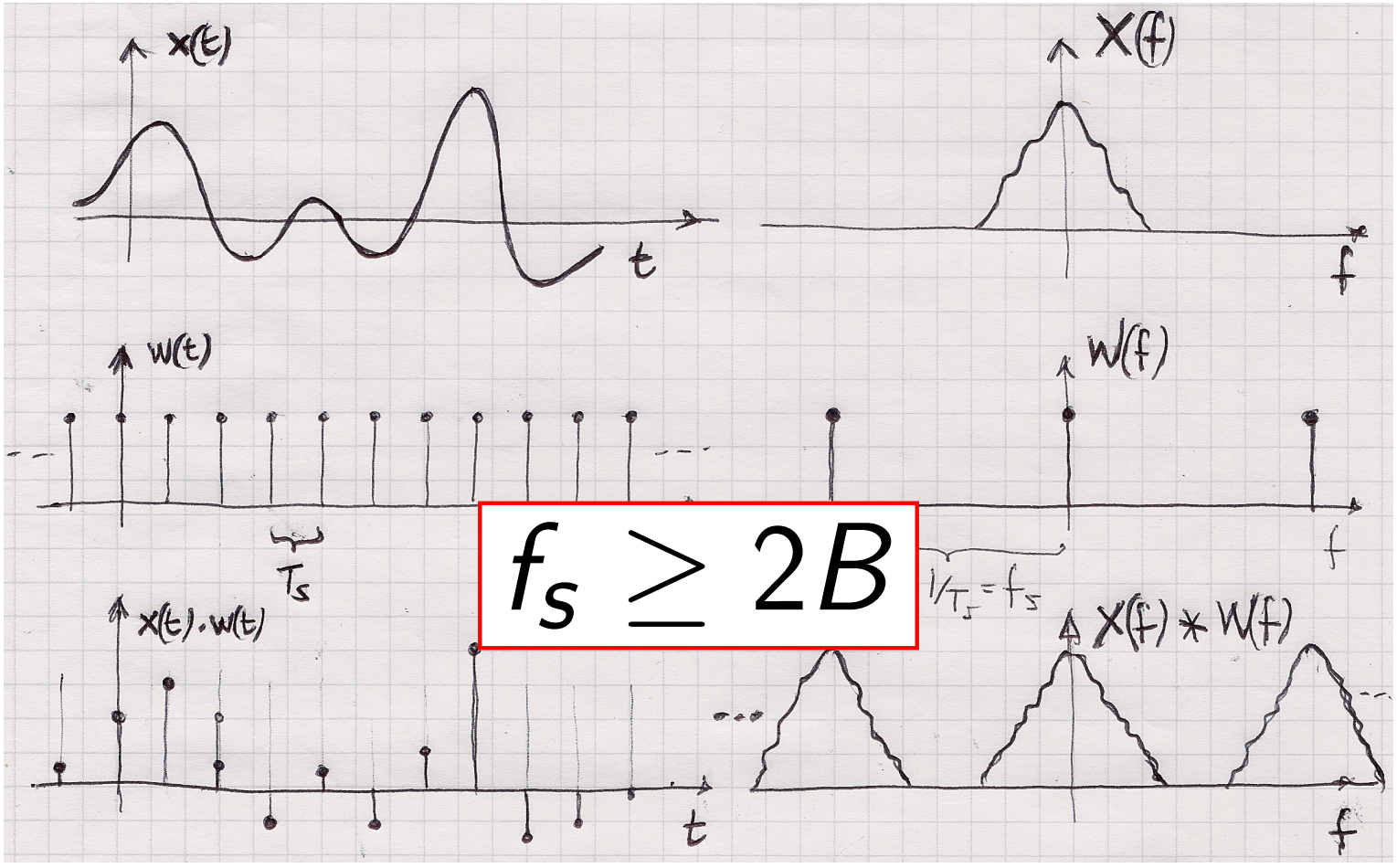


# Properties of Fourier Transform

| Time domain | | Frequency domain |
|---|---|---|
| $x[n]w[n]$ | $\Longleftrightarrow$ | $X(\omega) * W(\omega)$ |

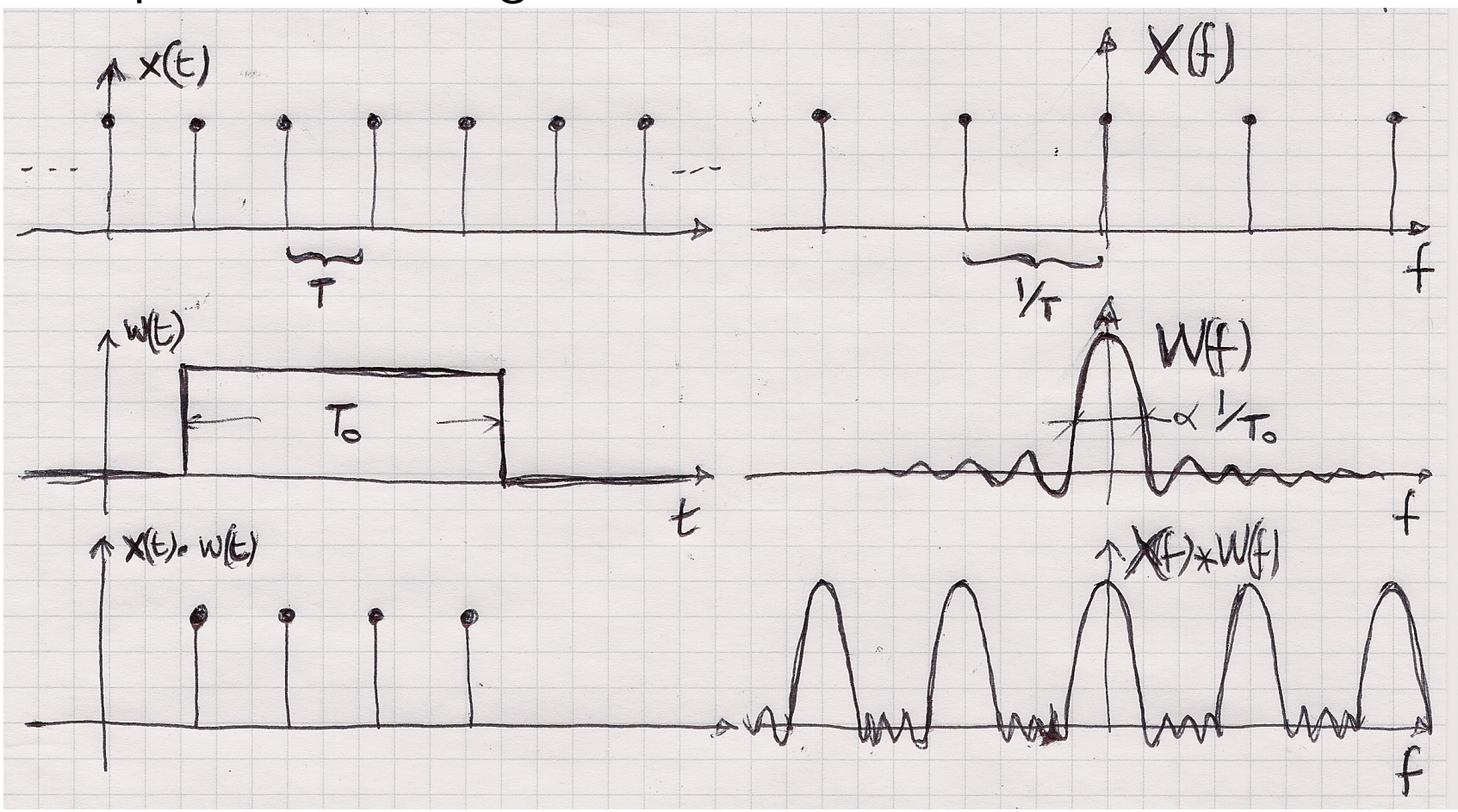
Example 1: sampling



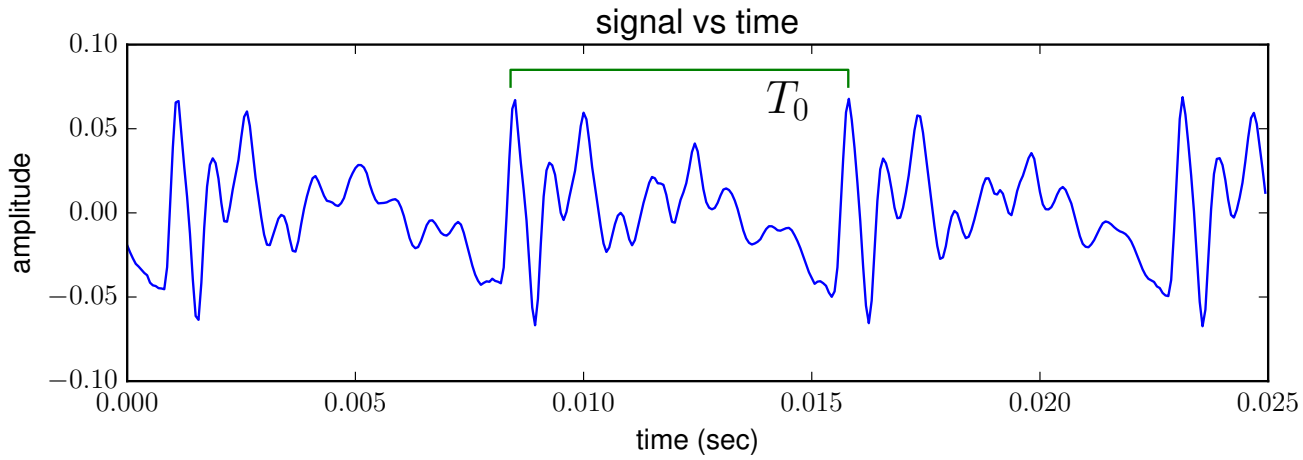
# Properties of Fourier Transform

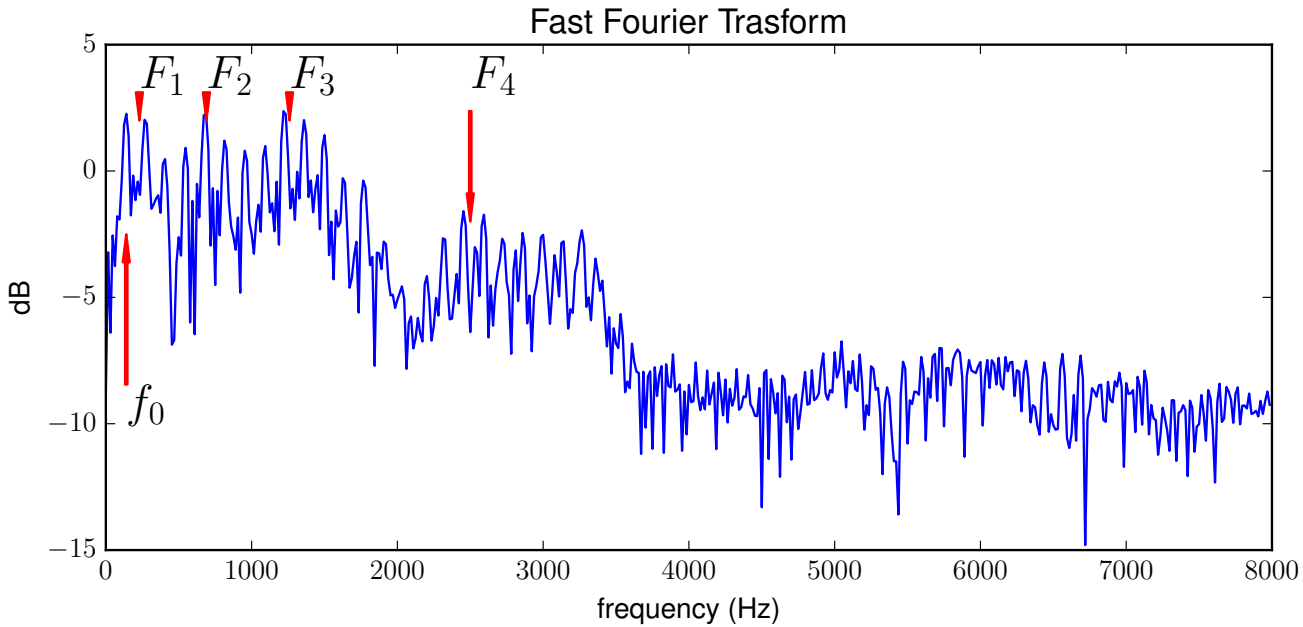
| Time domain | | Frequency domain |
|---|---|---|
| $x[n]w[n]$ | $\Longleftrightarrow$ | $X(\omega) * W(\omega)$ |

Example 2: windowing



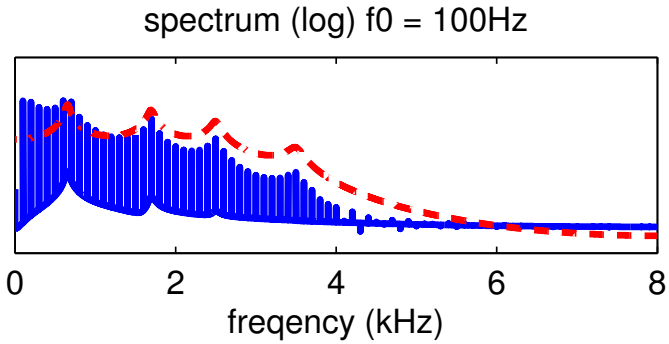
# Example: vowel

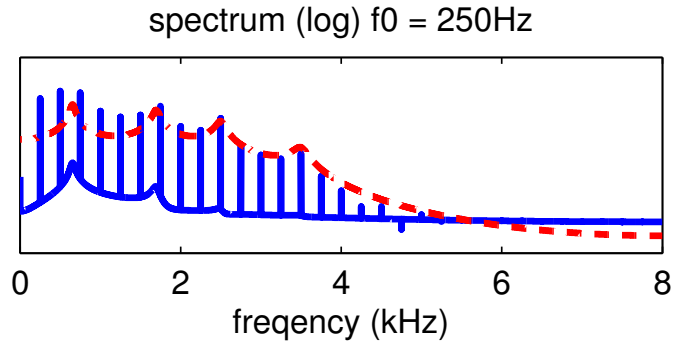


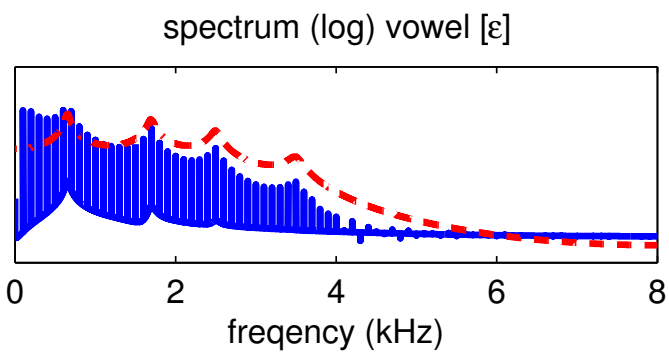
# $F_0$ and Formants

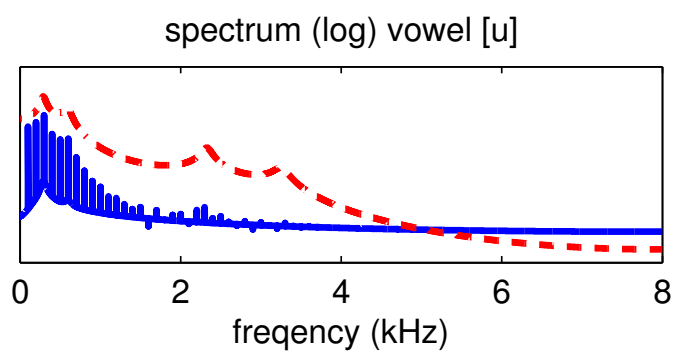
- Varying $F_0$ (vocal fold oscillation rate)




- Varying Formants (vocal tract shape)

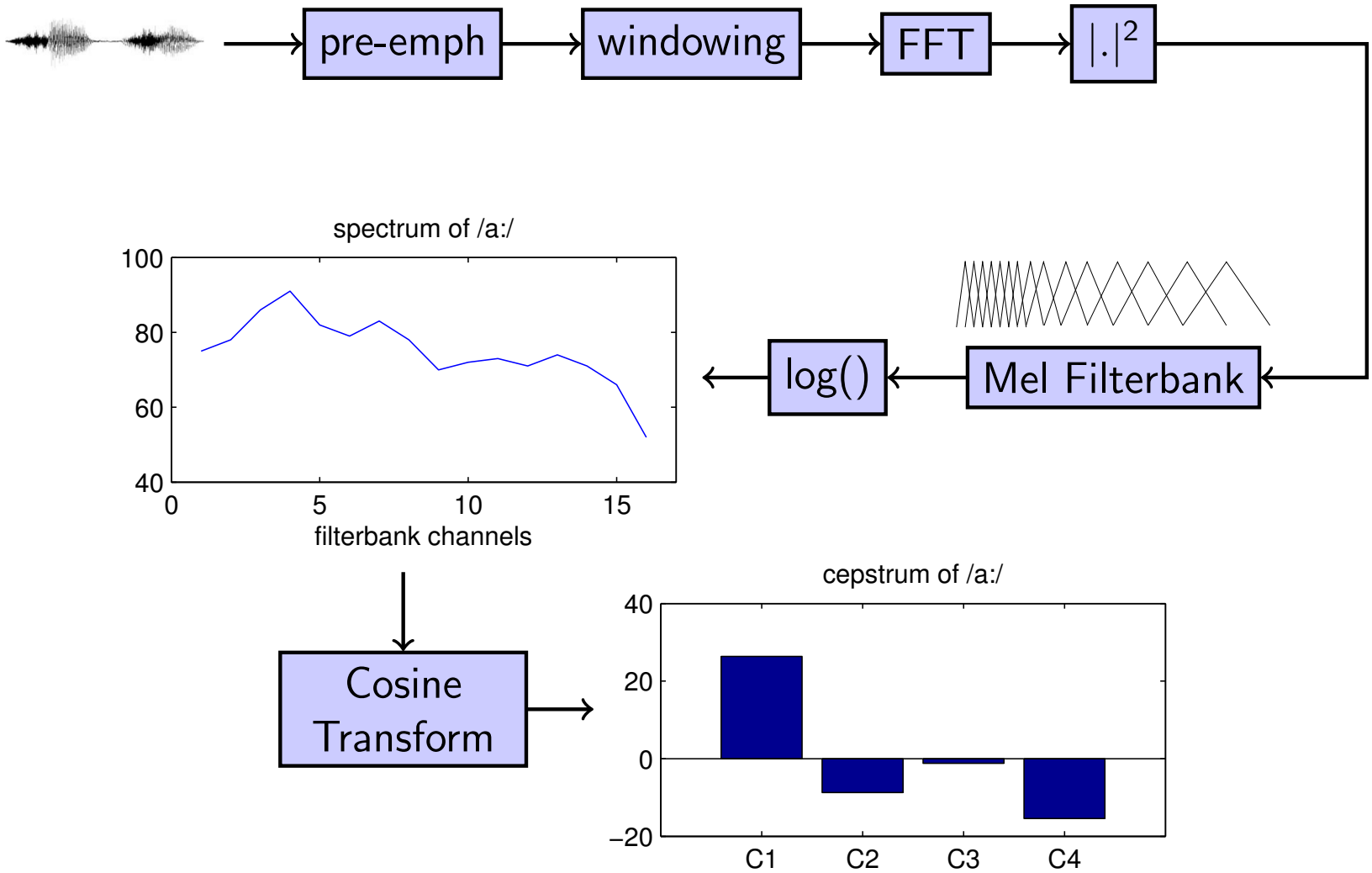


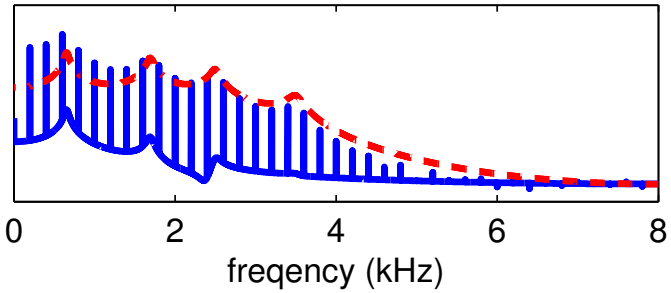
# Mel Frequency Cepstrum Coefficients

- *de facto* standard in ASR (before Deep Learning)
- imitate aspects of auditory processing
- does not assume all-pole model of the spectrum
- uncorrelated: easier to model statistically

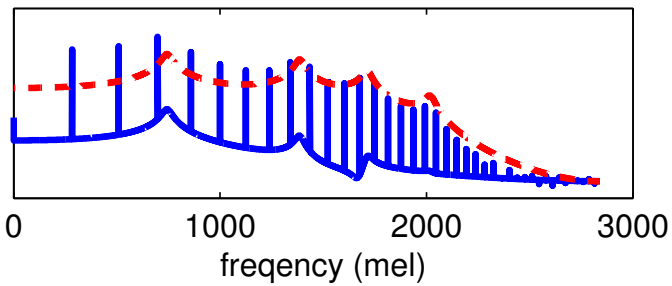
# MFCCs Calculation



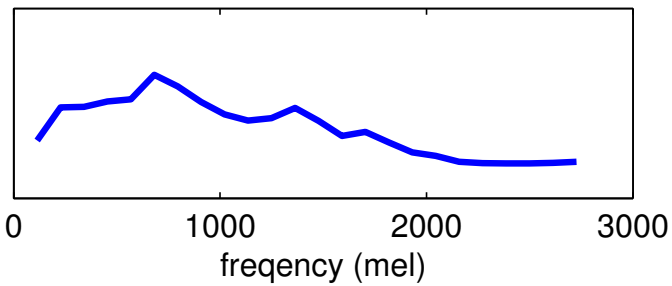
# Mel Frequency Cepstral Coefficients



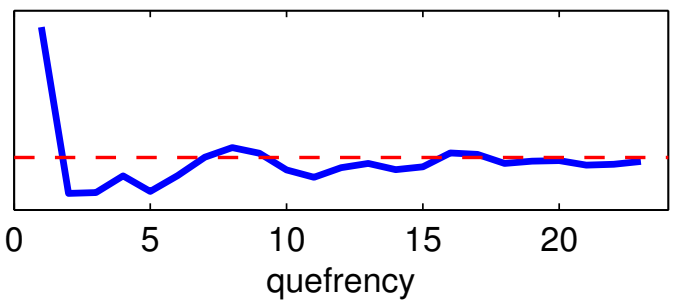

Linear to Mel frequency




Filterbank ($\sim$ 20-25 filters) + log()

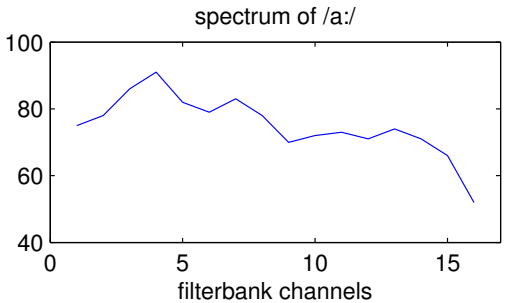



Discrete Cosine Transform

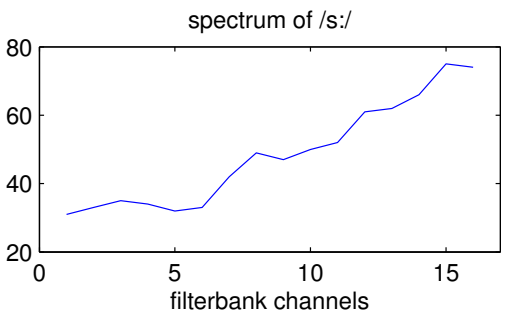


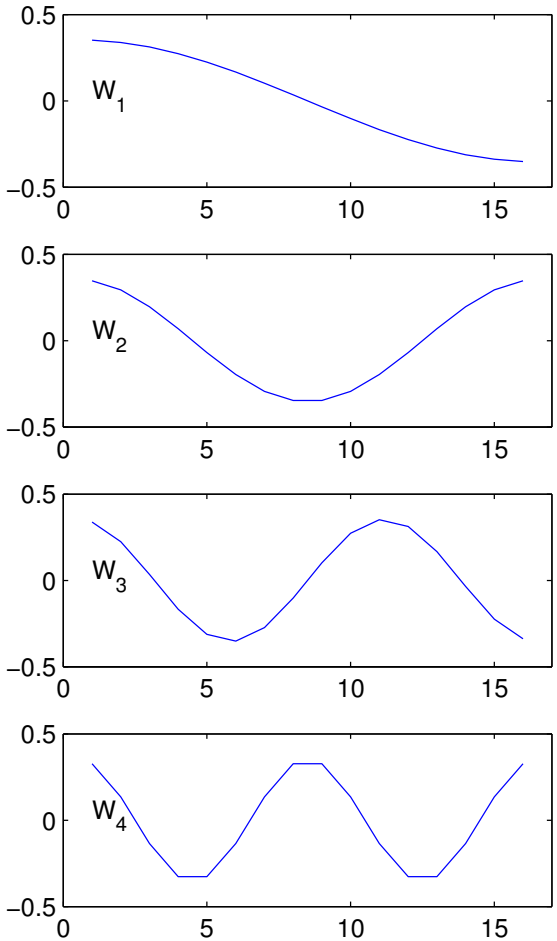
# MFCC: Cosine Transform

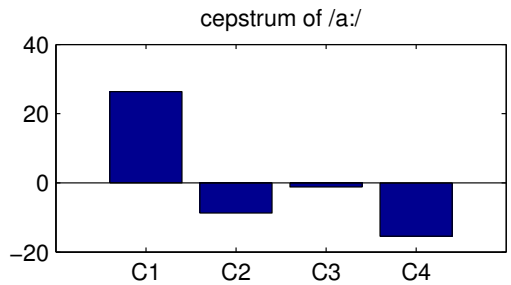
$$C_j = \sqrt{\frac{2}{N}} \sum_{i=1}^{N} A_i \cos\left(\frac{j\pi(i-0.5)}{N}\right)$$



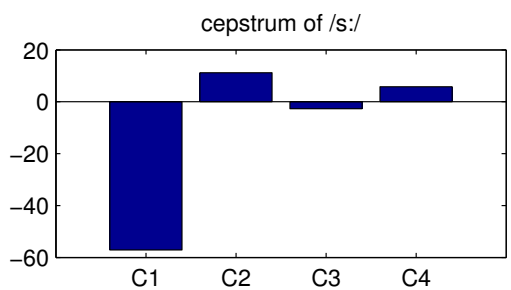

$A_i$




$C_j$



# MFCC Rationale

- signals combined in a convolutive way: $a[n] * b[n] * c[n]$
- in the spectral domain: $A(z)B(z)C(z)$
- taking the log: $\log(A(z)) + \log(B(z)) + \log(C(z))$
- to analise the different contribution perform Fourier transform (DCT if not interested in phase information).
- Terminology:
  - frequency vs quefrency
  - spectrum vs cepstrum
  - filter vs lifter
  - ...

# MFCC Advantages [1]

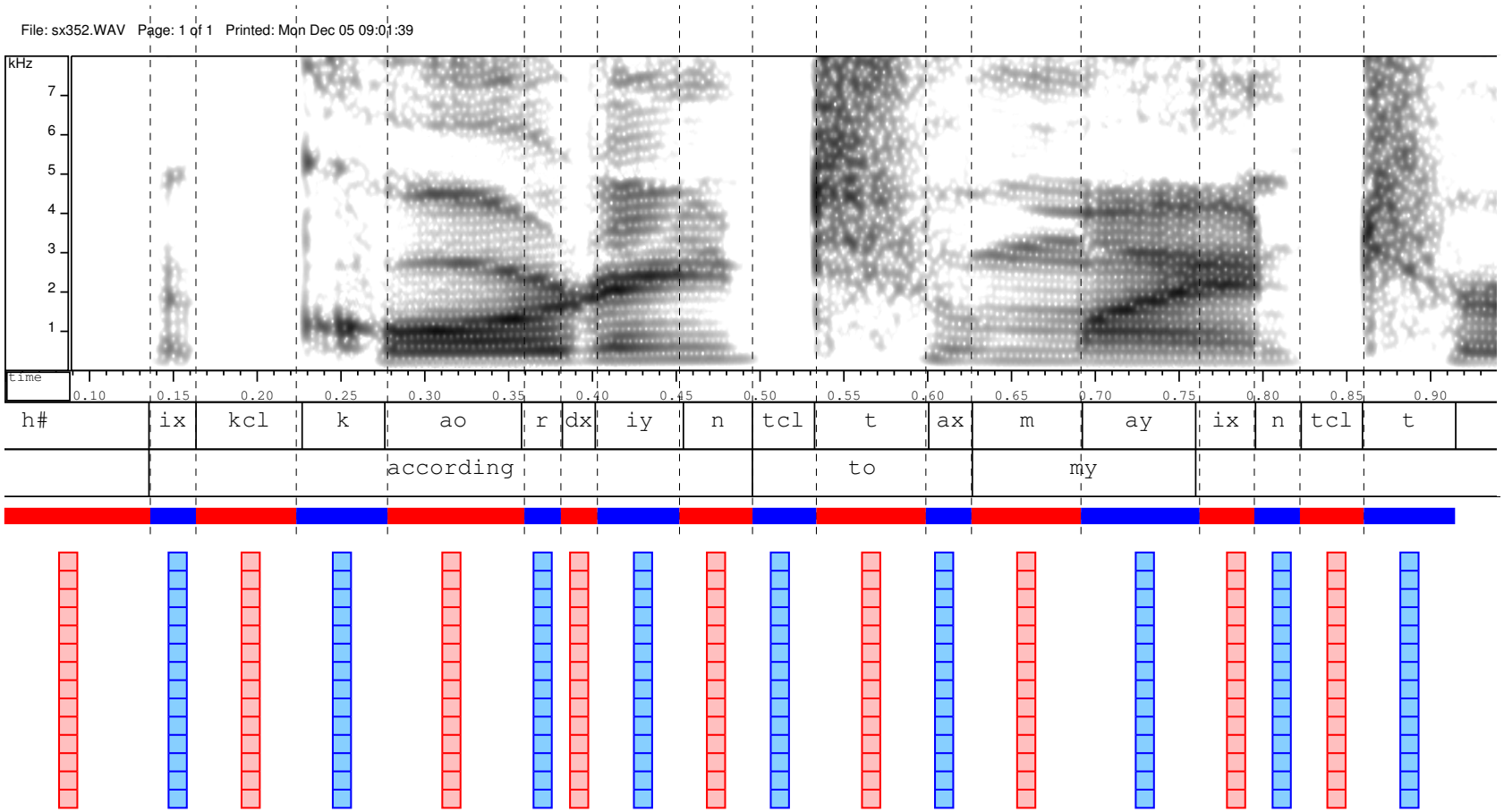
- fairly uncorrelated coefficients (simpler statistical models)
- high phonetic discrimination (empirically shown)
- do not assume all-pole model
- low number of coeff. enough to capture coarse structure of spectrum
- Cepstral Mean Subtraction corresponds to channel removal

[1] B. Bogert, M. Healy, and J. Tukey. "The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo-autocovariance, Cross-Cepstrum and Saphe Cracking". In: *Proc. Symp. Time Series Analysis*. Ed. by M. Rosemblatt. John Wiley & Sons, 1963, pp. 209–243

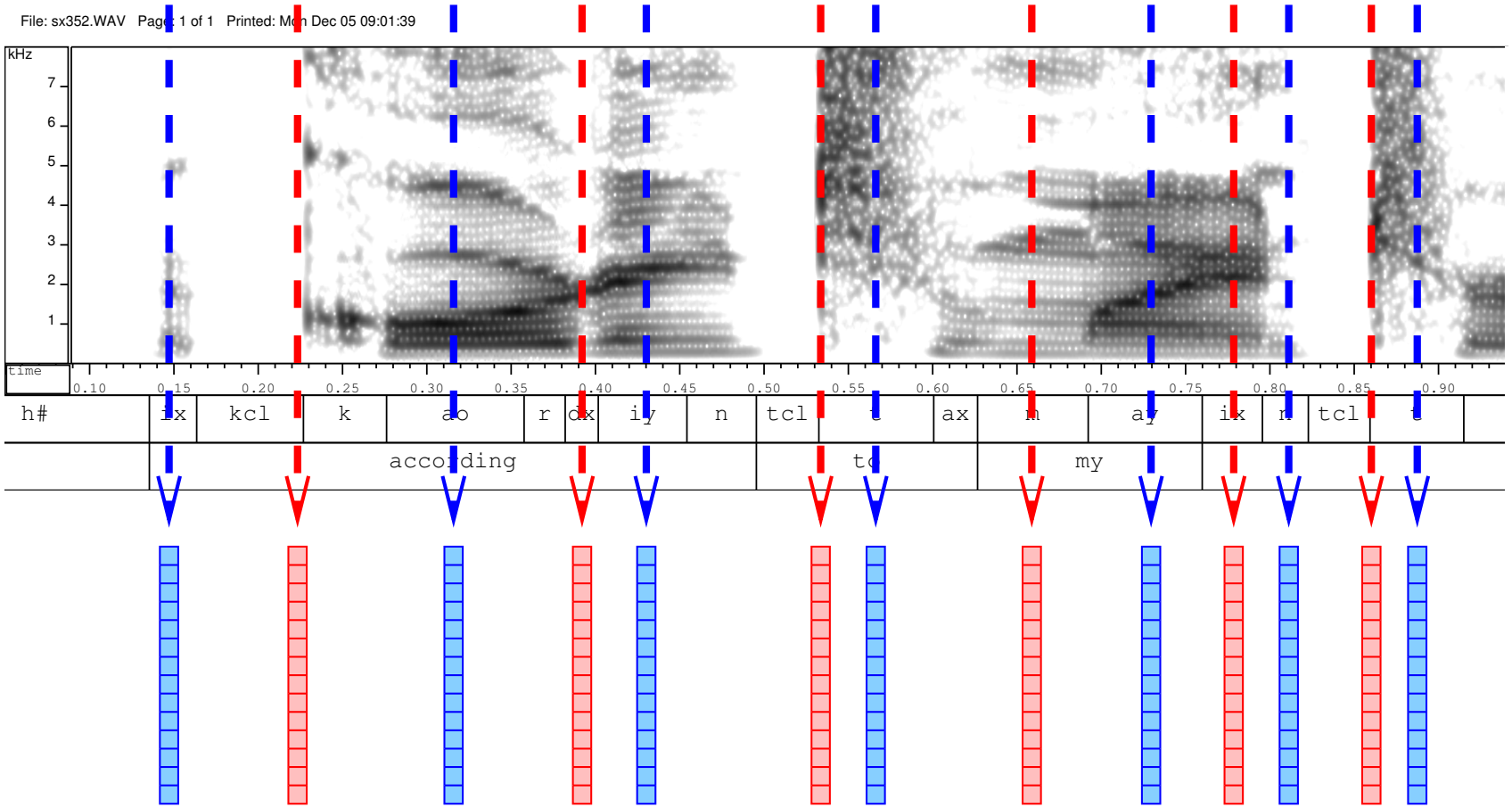
# MFCCs: typical values

- 12 Coefficients C1–C12
- Energy (could be C0)
- Delta coefficients (derivatives in time)
- Delta-delta (second order derivatives)
- total: 39 coefficients per frame (analysis window)

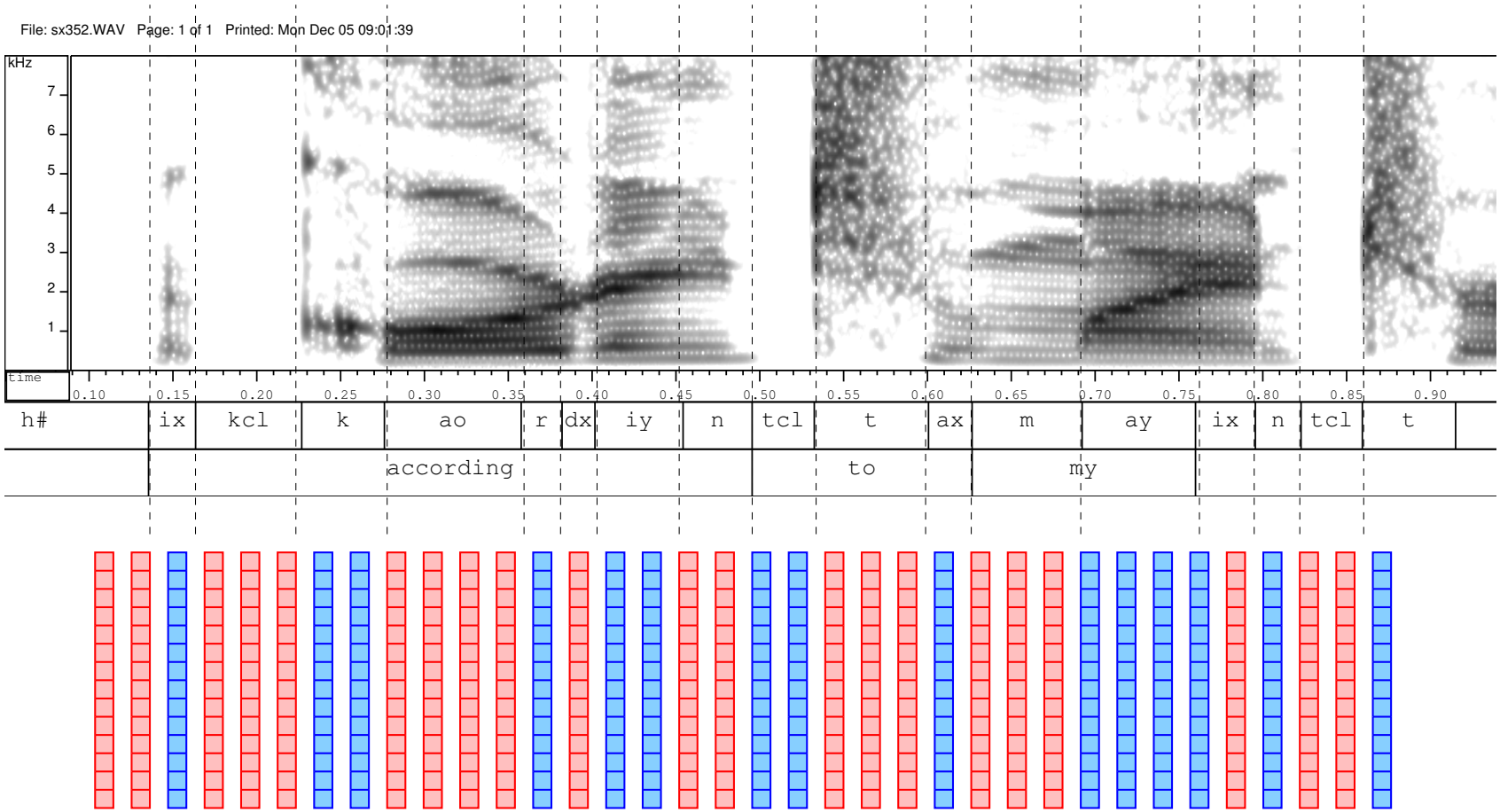
# Segment-Based Processing



# Landmark-Based Processing



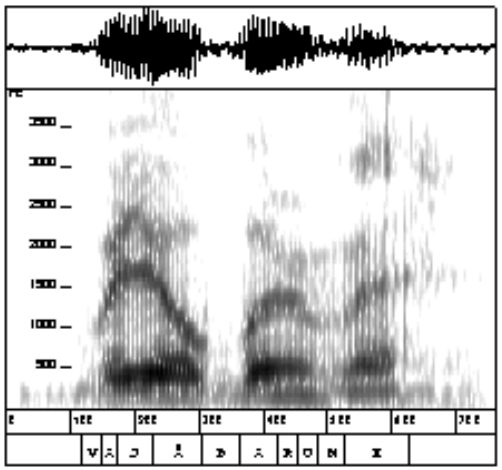
# Frame-Based Processing



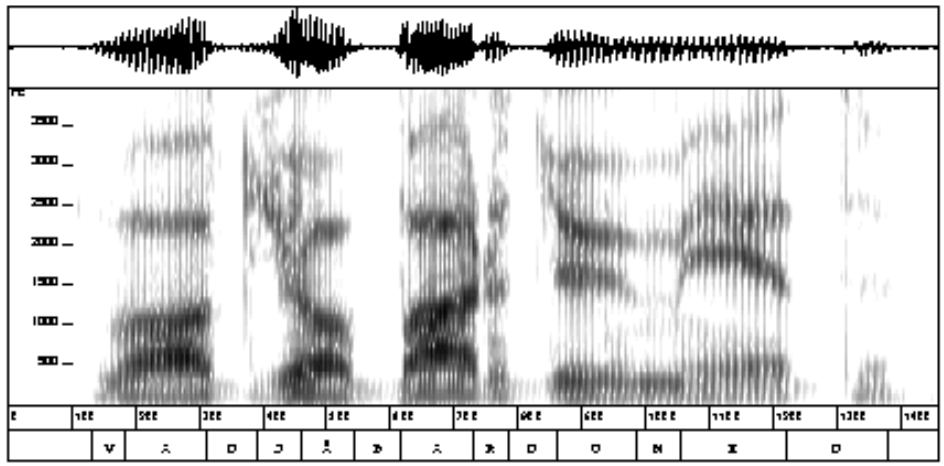
# Frame-wise distance metrics

| distance | $d(x, y)$ |
|---|---|
| city block: | $\sum_i \lvert x_i - y_i \rvert$ |
| Euclidean: | $\sqrt{\sum_i (x_i - y_i)^2}$ |
| Mahalanobis: | $\sum_i (x_i - \mu_y)^2 / \sigma_y$ |
| probability function: | $f(X = x \lvert \mu_y, \Sigma_y)$ |
| artificial neural networks: | $f\left(\sum_i w_i x_i - \theta\right)$ |

# Comparing Utterances



Va jobbaru me



Vad jobbar du med

"What is your occupation"
("What work you with")

# Combining frame-wise scores into utterance scores

Template Matching

- oldest technique
- simple comparison of template patterns
- compensate for varying speech rate (Dynamic Programming)

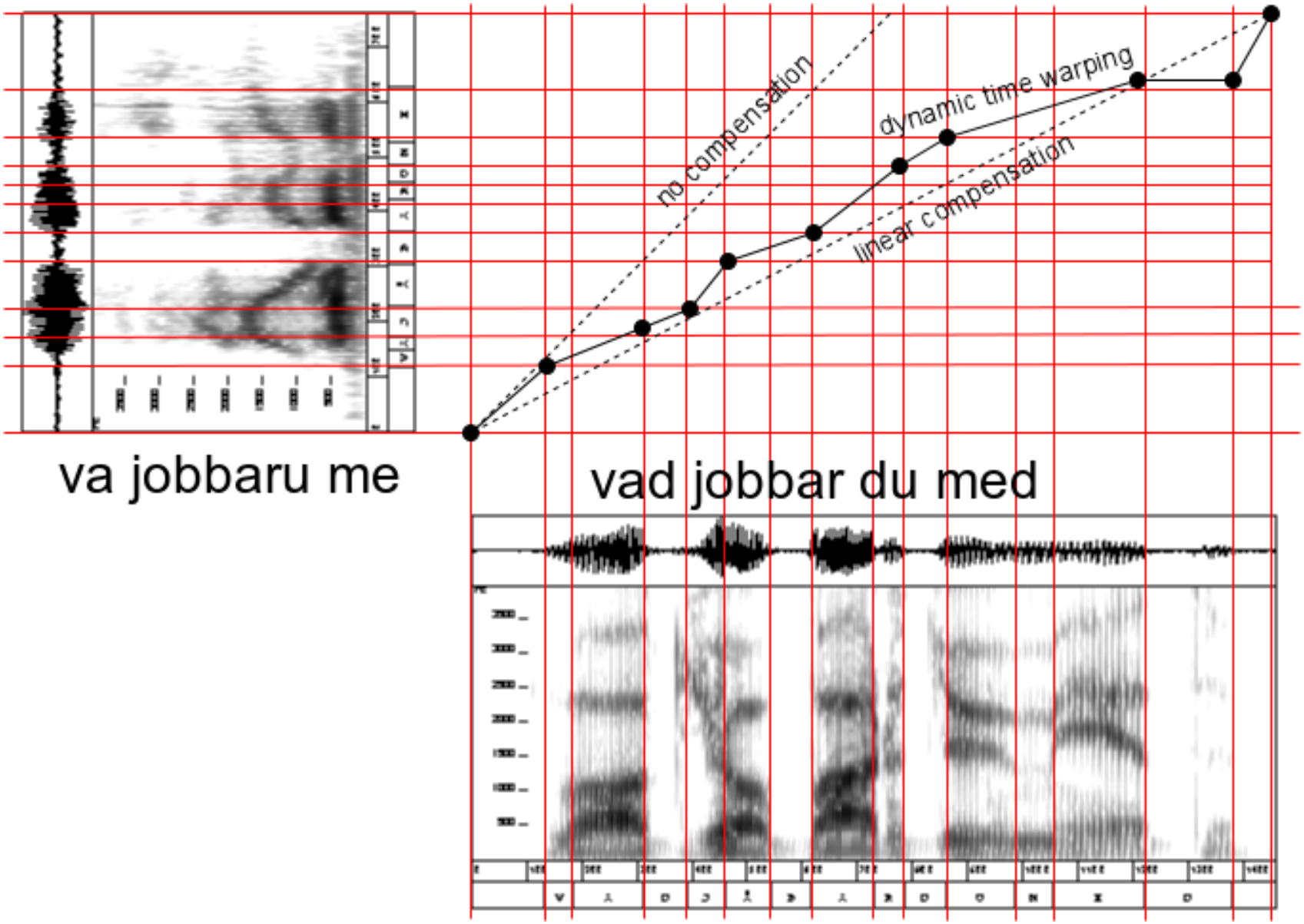
Hidden Markov Models (HMMs)

- most used technique
- models of segmental structure of speech
- recognition by Viterbi search (Dynamic Programming)

# Template Matching: Why?

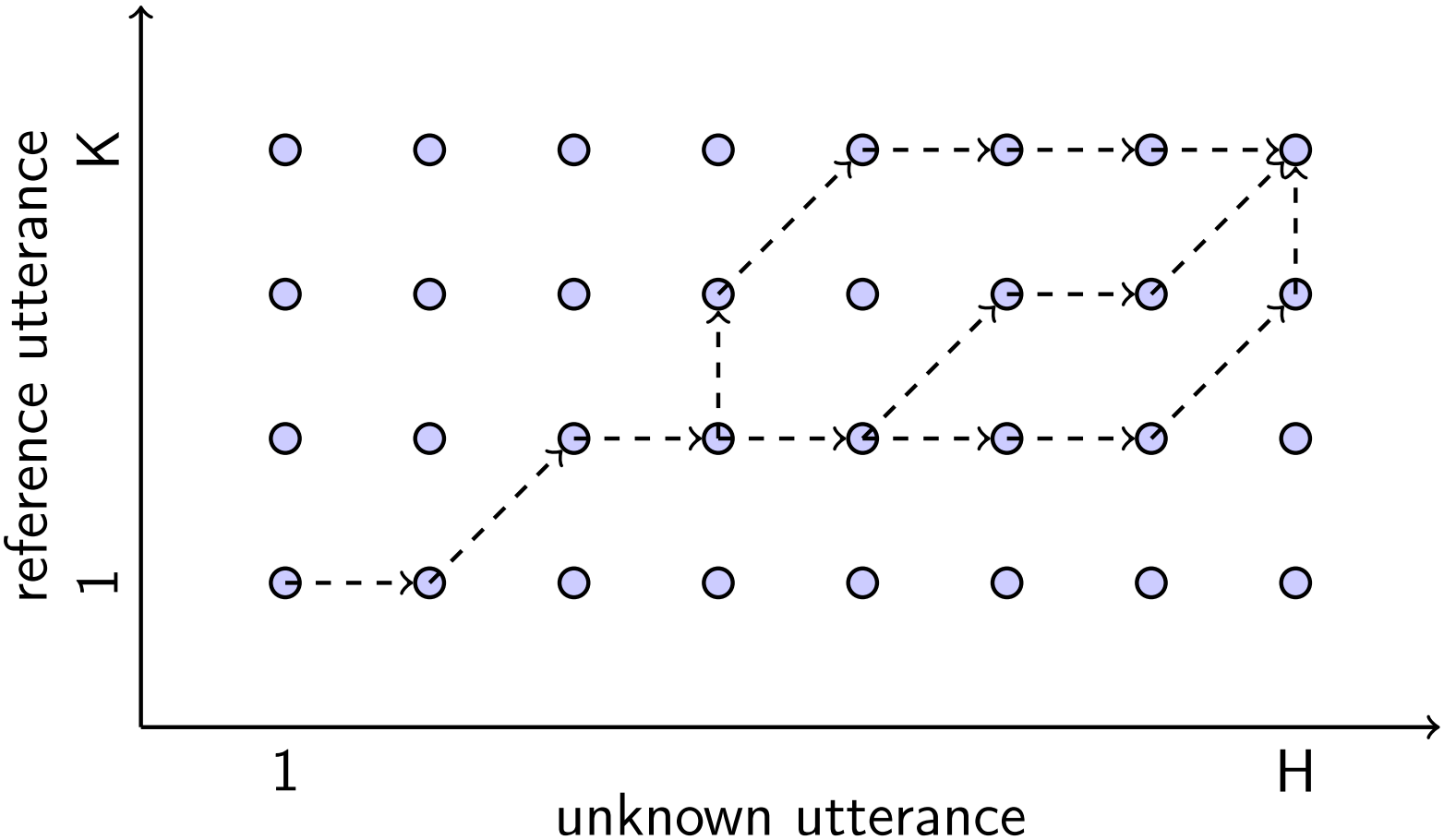
1. Historical: first technique used in ASR
2. Pedagogical: explain the problem and Dynamic Programming
3. Also called Dynamic Time Warping (DTW)

# Template Matching



# Dynamic Programming

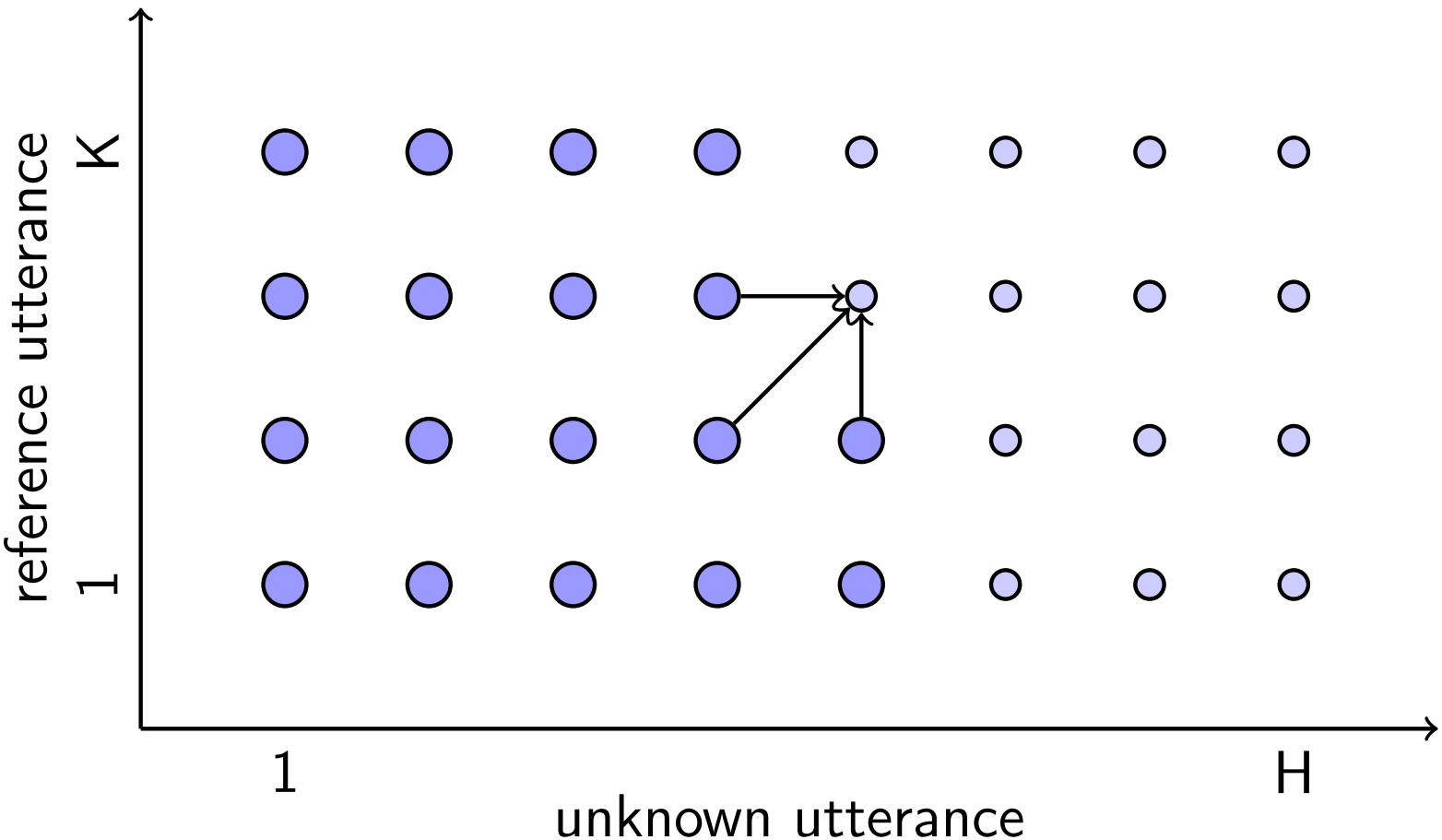
- compare any possible alignment
- problem: exponential with H and K!



# Dynamic Programming

Dynamic Time Warping (DTW) algorithm

1: **for** $h = 1$ to $H$ **do**
2: **for** $k = 1$ to $K$ **do**
3: $AccD[h, k] = LocD[h, k] + \min(AccD[h-1, k], AccD[h-1, k-1], AccD[h, k-1])$



# DP Example: Spelling

- observations are letters
- local distance: 0 (same letter), 1 (different letter)
- Unknown utterance: ALLDRIG
- Reference1: ALDRIG
- Reference2: ALLTID
- Problem: find closest match

Distance char-by-char:

- ALLDRIG–ALDRIG = 5
- ALLDRIG–ALLTID = 4

# DP Example: Solution

*LocD*[h,k]=

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| G | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| I | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| R | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| D | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| L | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| A | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| | A | L | L | D | R | I | G |

*AccD*[h,k]=

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| G | 5 | 4 | 4 | 3 | 2 | 1 | 0 |
| I | 4 | 3 | 3 | 2 | 1 | 0 | 1 |
| R | 3 | 2 | 2 | 1 | 0 | 1 | 2 |
| D | 2 | 1 | 1 | 0 | 1 | 2 | 3 |
| L | 1 | 0 | 0 | 1 | 2 | 3 | 4 |
| A | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| | A | L | L | D | R | I | G |

Distance ALLDRIG–ALDRIG: *AccD*[H,K] = 0

Distance ALLDRIG–ALLTID? (5min)

# DP Example: Solution

*LocD*[h,k]=

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| D | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| I | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| T | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| L | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| L | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| A | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| | A | L | L | D | R | I | G |

*AccD*[h,k]=

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| D | 5 | 3 | 3 | 2 | 3 | 3 | 3 |
| I | 4 | 2 | 2 | 2 | 2 | 2 | 3 |
| T | 3 | 1 | 1 | 1 | 2 | 3 | 4 |
| L | 2 | 0 | 0 | 1 | 2 | 3 | 4 |
| L | 1 | 0 | 0 | 1 | 2 | 3 | 4 |
| A | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| | A | L | L | D | R | I | G |

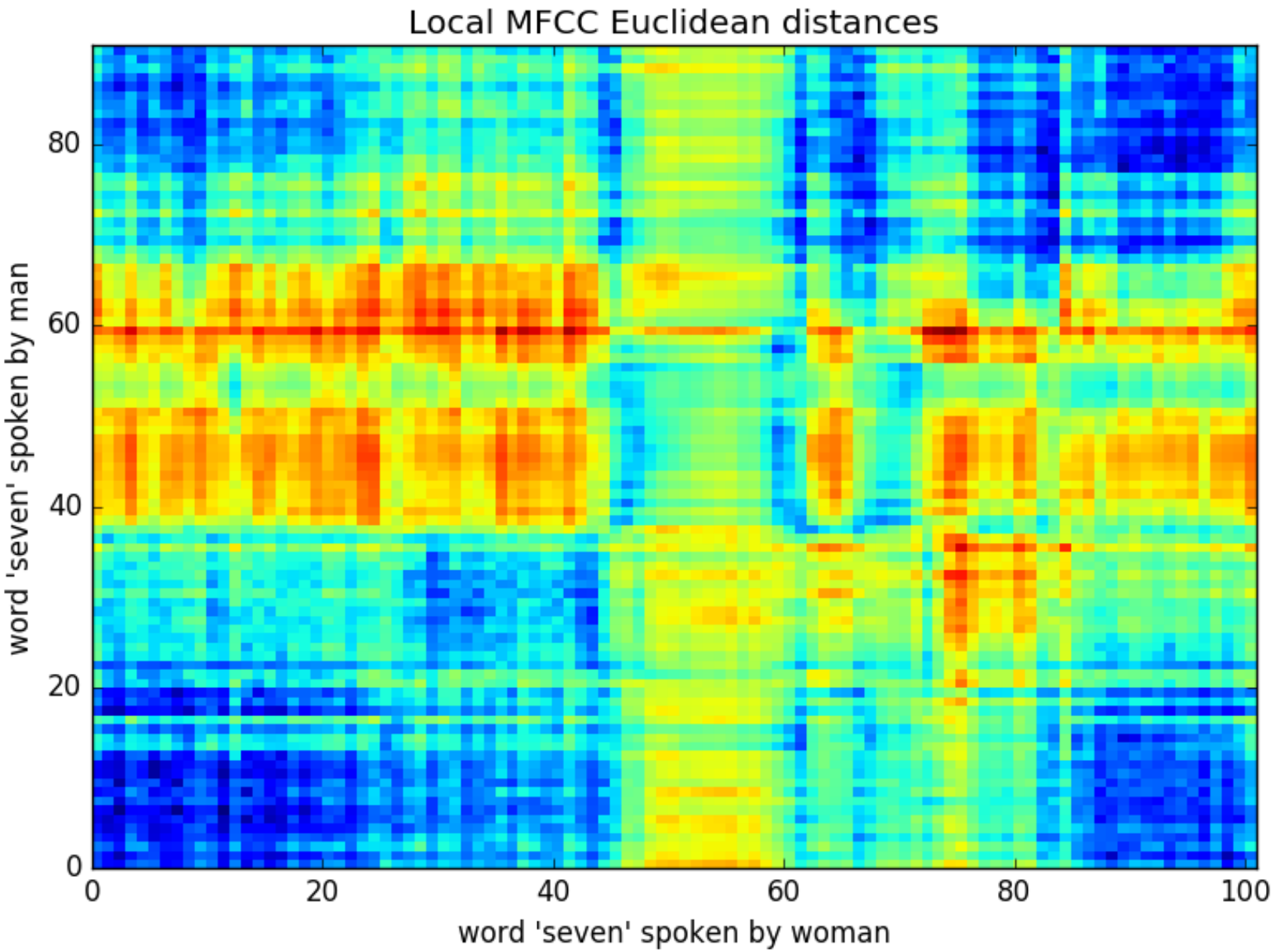
Distance ALLDRIG–ALDRIG: $AccD[H,K] = 0$
Distance ALLDRIG–ALLTID: $AccD[H,K] = 3$

# Best path: Backtracking

Sometimes we want to know the path

1. at each point [h,k] remember the minimum distance predecessor (back pointer)
2. at the end point [H,K] follow the back pointers until the start

# Real Example from Lab1



# Properties of Template Matching

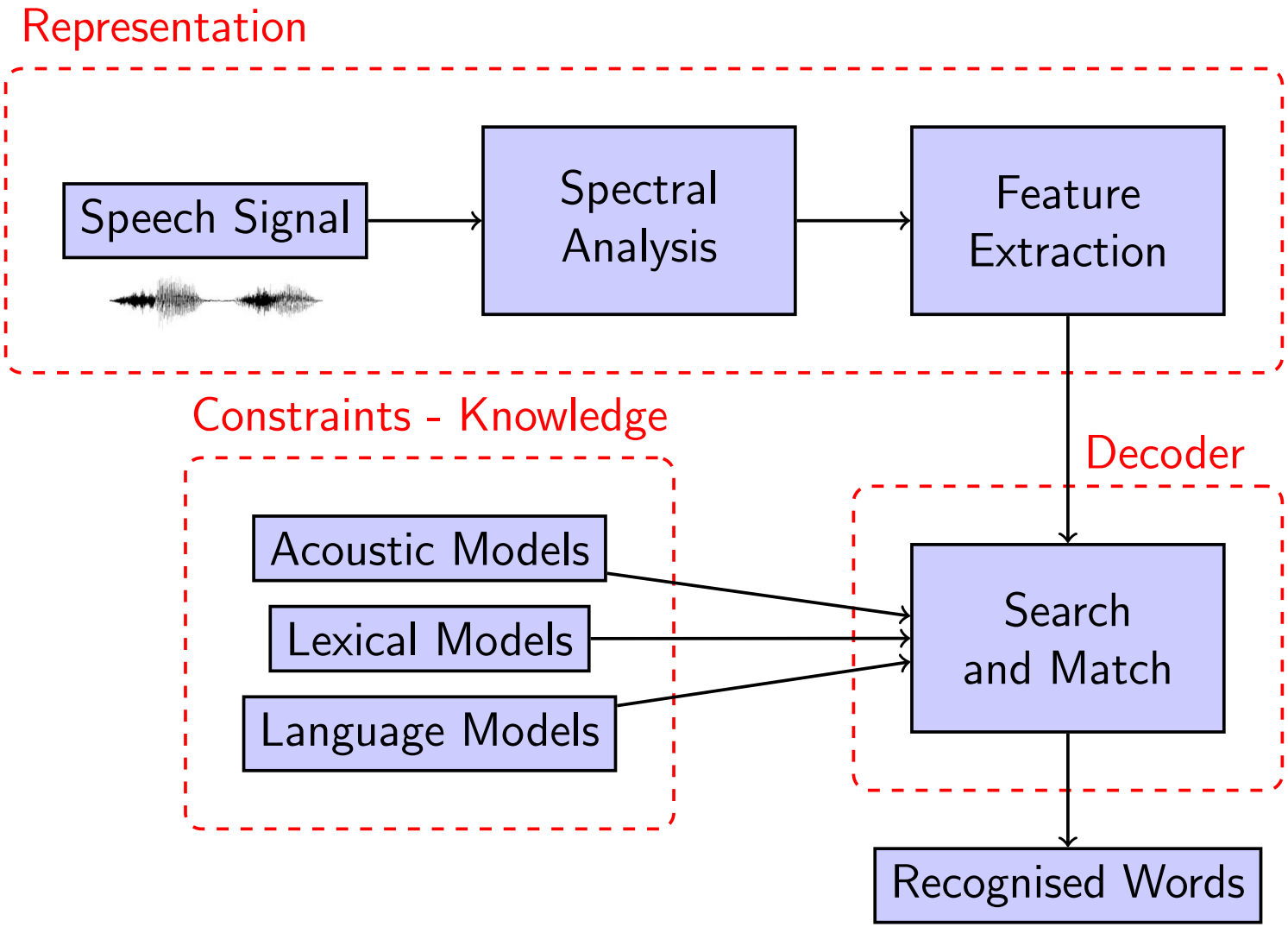
Pros:

+ No need for phonetic transcriptions
+ within-word co-articulation for free
+ high time resolution

Cons:

– cross-word co-articulation not modelled
– needs word segmentation (isolated words)
– requires recordings of every word
– not easy to model variation
– does not scale up with vocabulary size

# Components of ASR System



# Assuming you know:

- Probability theory
- Basic probability distributions
- Basic machine learning

# Bayes' Rule

if

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

then

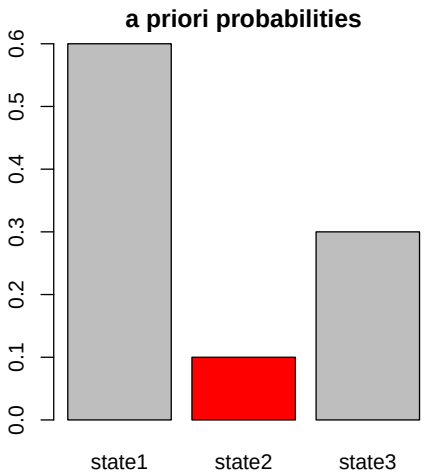
$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

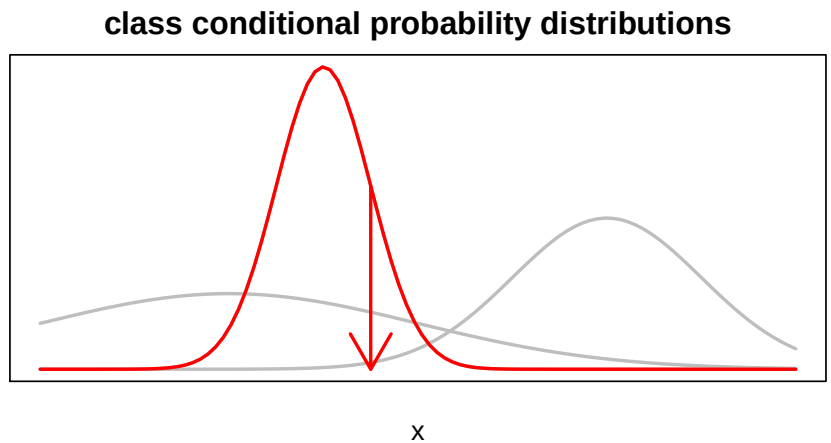
and

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

# The Probabilistic Model of Classification

- "Nature" assumes one of $c$ states $\omega_j$ with *a priori* probability $P(\omega_j)$
- When in state $\omega_j$, "nature" emits observations $\hat{\mathbf{x}}$ with distribution $p(\mathbf{x}|\omega_j)$

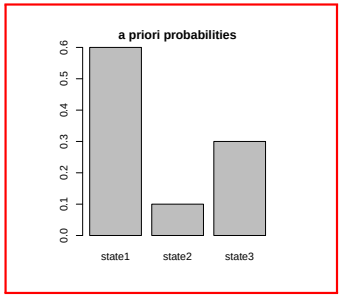


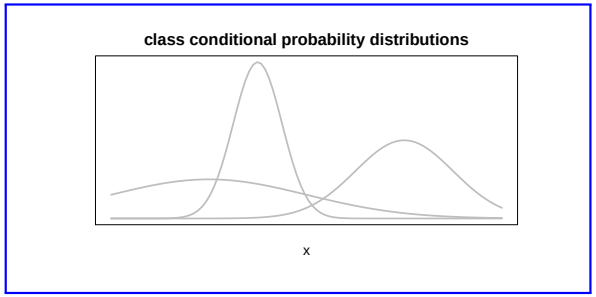



# Problem

- If I observe $\hat{\mathbf{x}}$ and I know $P(\omega_j)$ and $p(\mathbf{x}|\omega_j)$ for each $j$
- what can I say about the state of "nature" $\omega_j$?

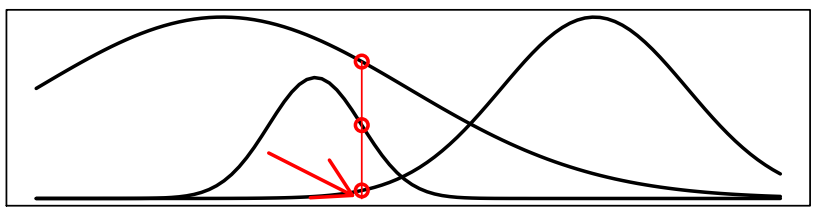
# Bayes decision theory



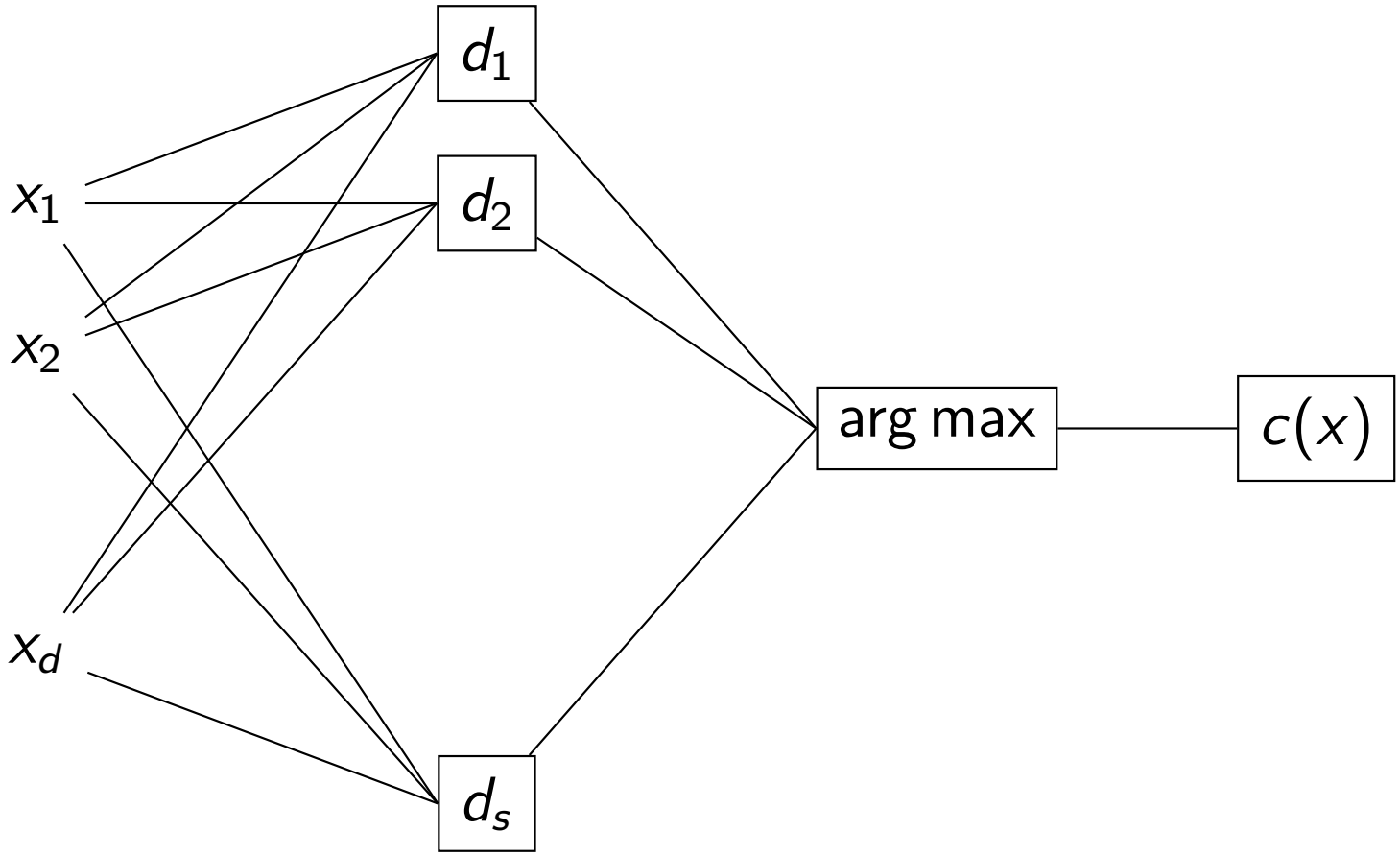




$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)\ P(\omega_j)}{p(\mathbf{x})}$$

**posterior probabilities**



# Classifiers: Discriminant Functions




$$d_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)\ P(\omega_i)$$

# Classifiers: Decision Boundaries

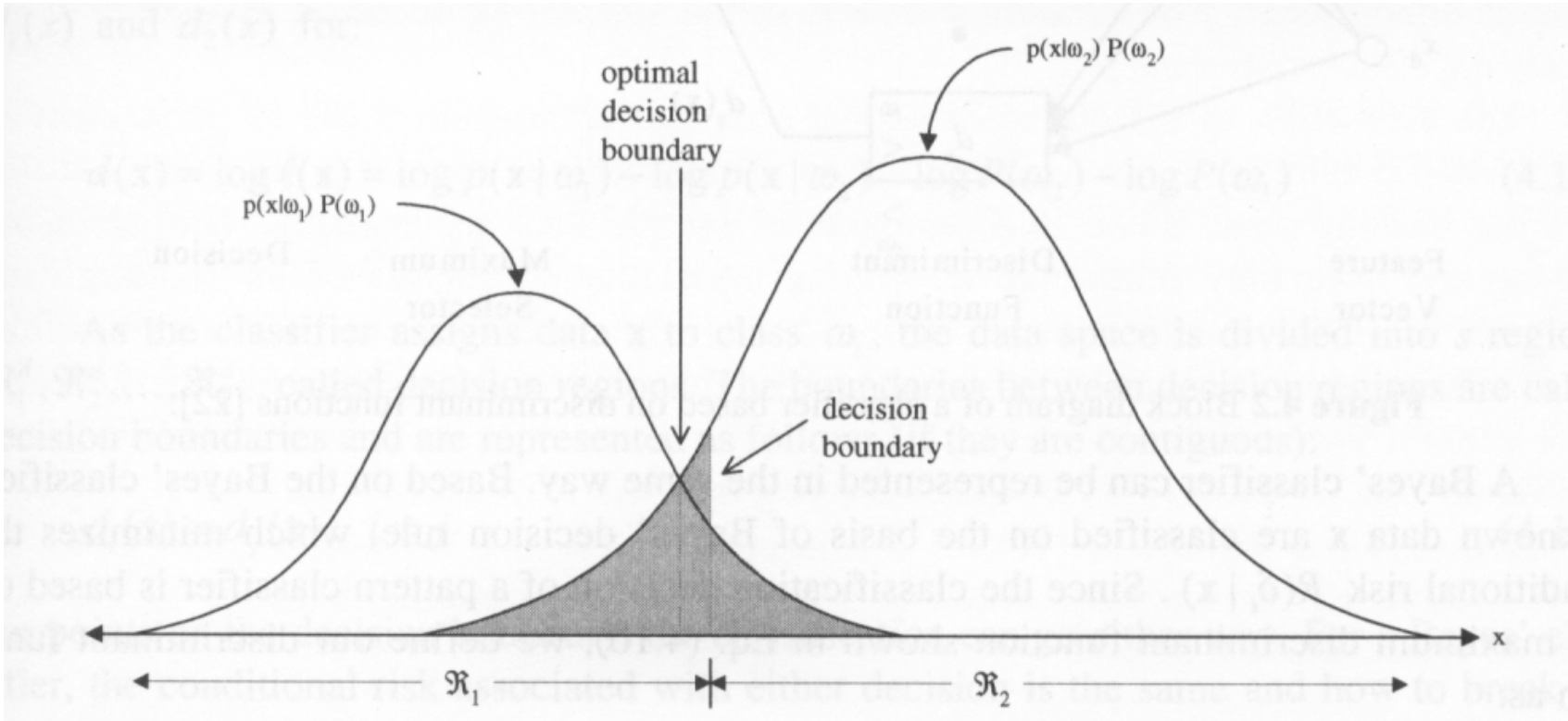



Figure from Huang, Acero, Hon.

# Decision Boundaries in Two Dimensions

https://github.com/giampierosalvi/GaussianDecisionBoundaries

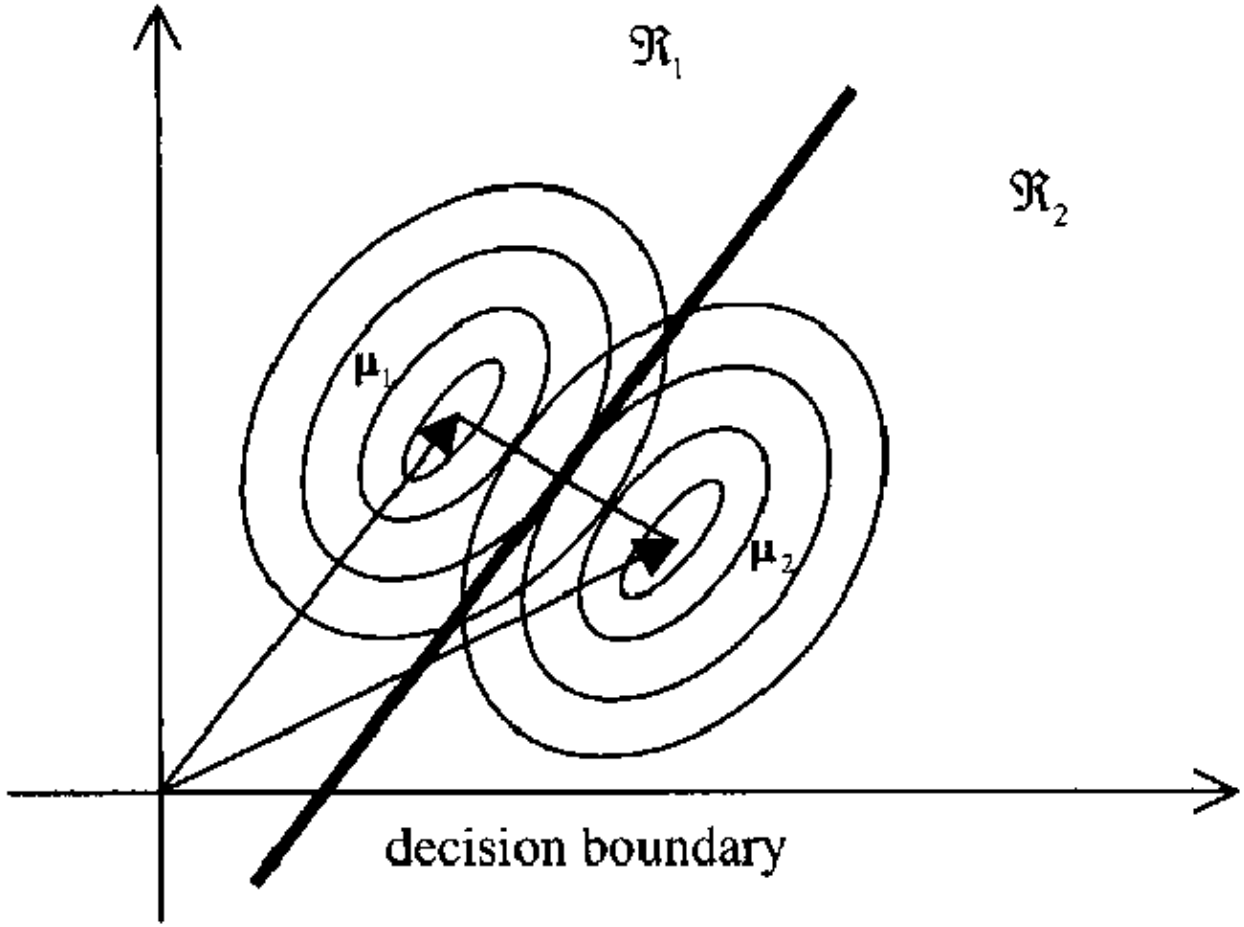



Figure from Huang, Acero, Hon.

# Bayes' Rule and Speech Recognition

$A$ = words, $B$ = sounds:

- During training we know the words and can compute $P(\text{sounds}|\text{words})$ using frequentist approach (repeated observations)
- during recognition we want $\widehat{\text{words}} = \arg\max P(\text{words}|\text{sounds})$
- using Bayes' rule:

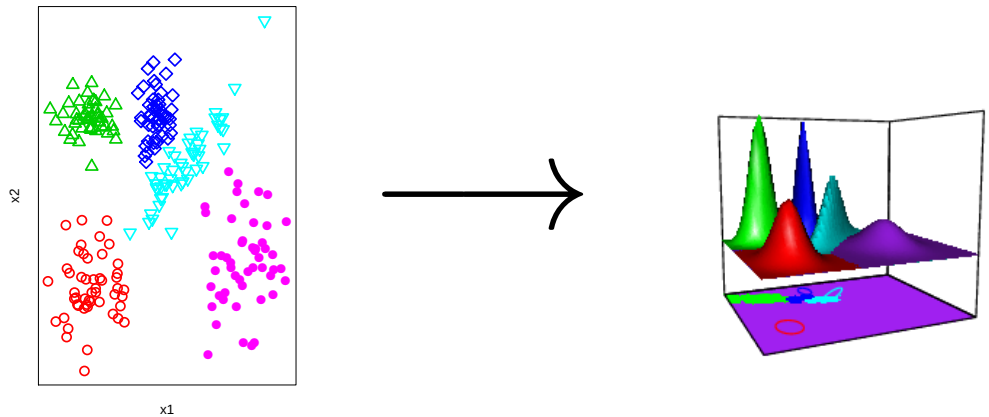
$$P(\text{words}|\text{sounds}) = \frac{P(\text{sounds}|\text{words})P(\text{words})}{P(\text{sounds})}$$

where
$P(\text{words})$: *a priori* probability of the words (Language Model)
$P(\text{sounds})$: *a priori* probability of the sounds (constant, can be ignored)

# Estimation Theory

- so far we assumed we know $P(\omega_j)$ and $p(\mathbf{x}|\omega_j)$
- how can we obtain them from collections of data?
- this is the subject of Estimation Theory

# Parameter estimation

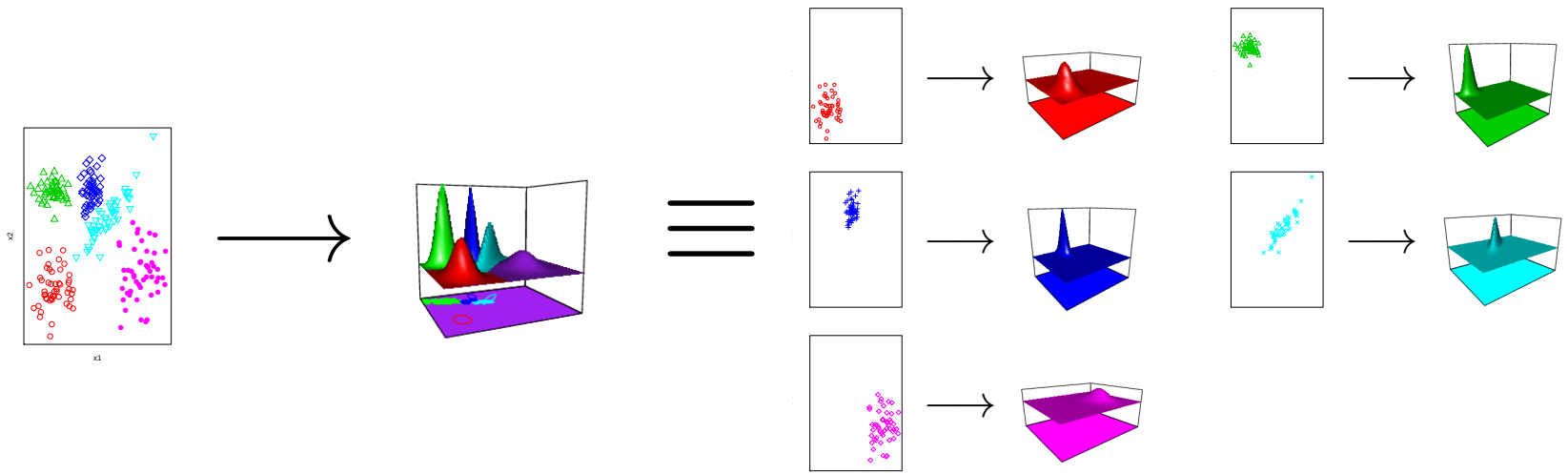



Assumptions:

- samples from class $\omega_i$ do not influence estimate for class $\omega_j$, $i \neq j$
- samples from the same class are independent and identically distributed (i.i.d.)

# Parameter estimation (cont.)

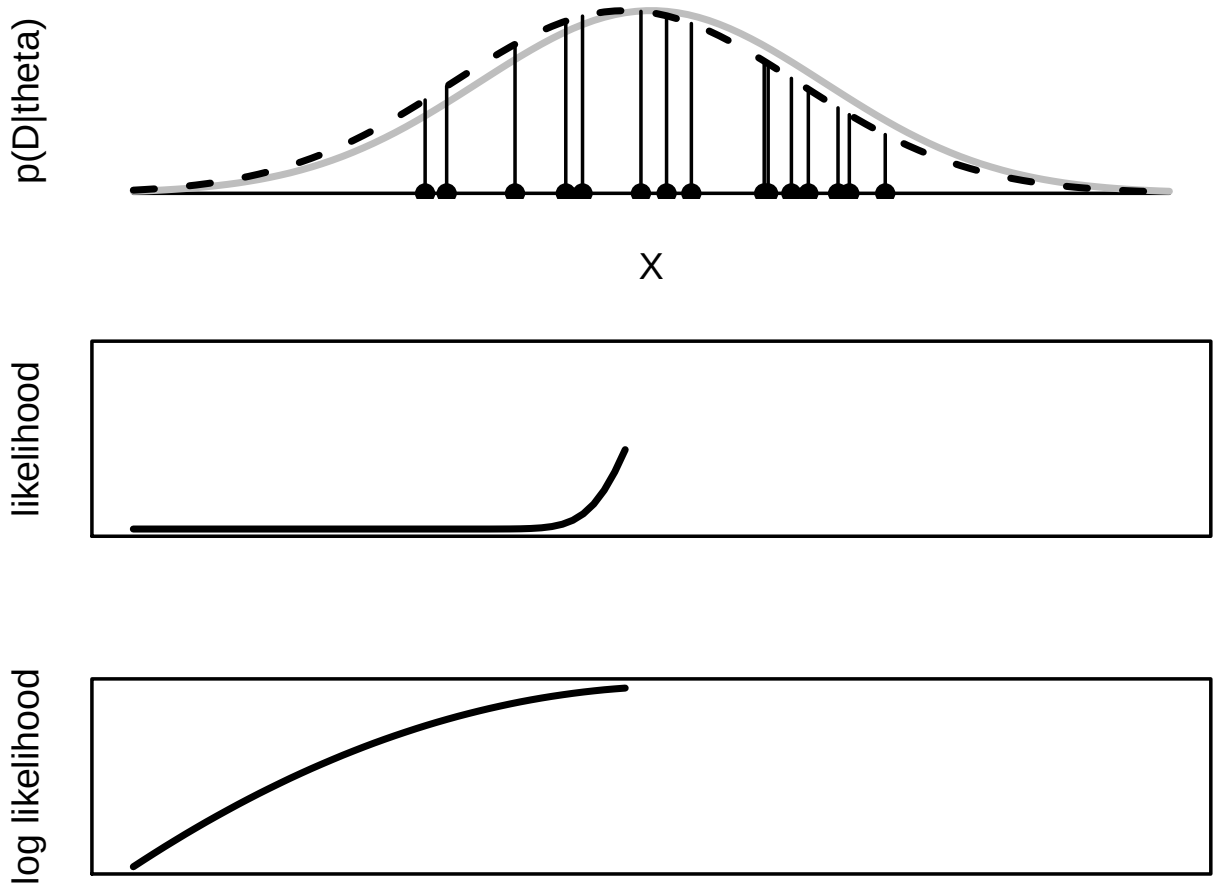
- class independence assumption:



- Maximum likelihood estimation
- Maximum a posteriori estimation
- Bayesian estimation

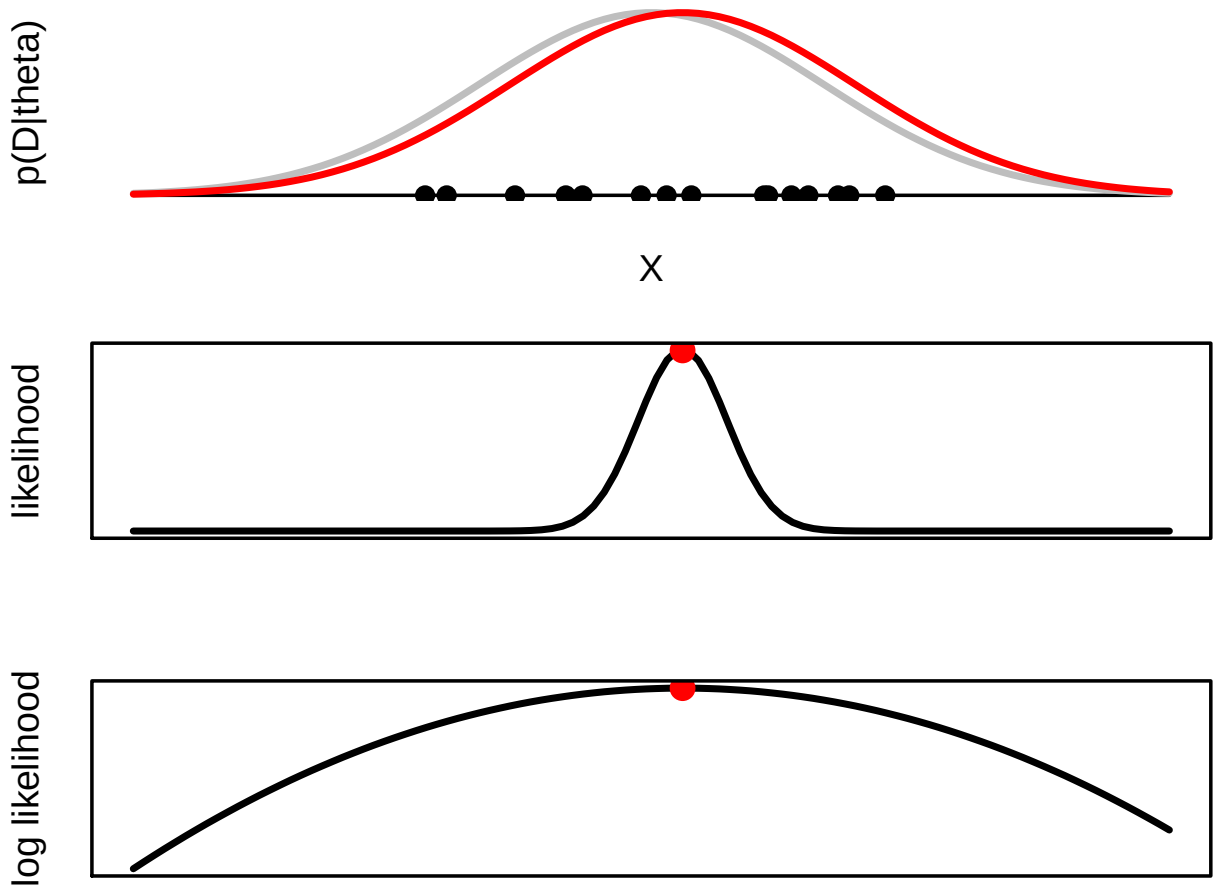
# Maximum likelihood estimation

- Find parameter vector $\hat{\theta}$ that maximises $p(\mathcal{D}|\theta)$ with $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
- i.i.d. $\rightarrow p(\mathcal{D}|\theta) = \prod_{k=1}^{n} p(\mathbf{x}_k|\theta)$



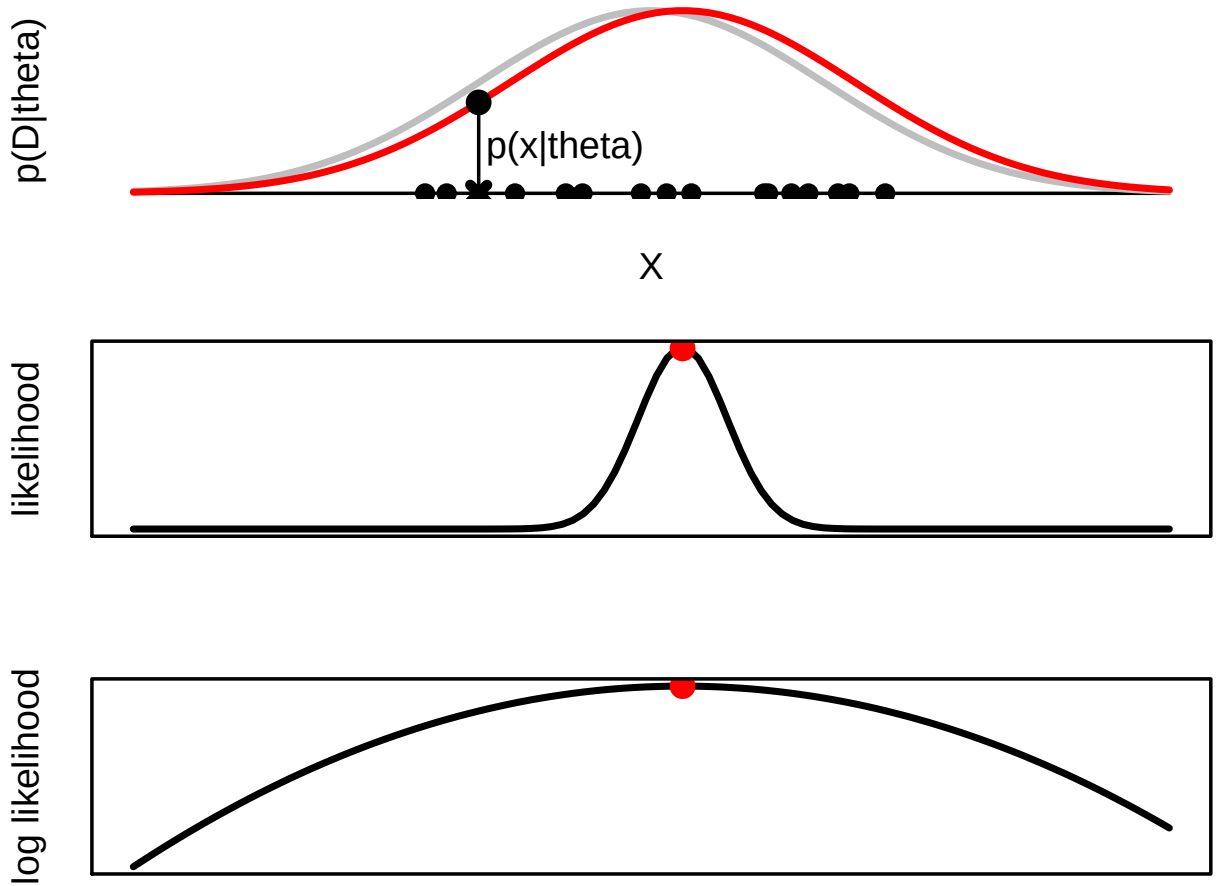
# Maximum likelihood estimation

- Find parameter vector $\hat{\theta}$ that maximises $p(\mathcal{D}|\theta)$ with $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
- i.i.d. $\rightarrow p(\mathcal{D}|\theta) = \prod_{k=1}^{n} p(\mathbf{x}_k|\theta)$



# Maximum likelihood estimation

- Find parameter vector $\hat{\theta}$ that maximises $p(\mathcal{D}|\theta)$ with $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
- i.i.d. $\rightarrow p(\mathcal{D}|\theta) = \prod_{k=1}^{n} p(\mathbf{x}_k|\theta)$



# ML estimation of Gaussian mean

$$N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \text{ with } \theta = \{\mu,\sigma^2\}$$

Log-likelihood of data (i.i.d. samples):

$$\log P(\mathcal{D}|\theta) = \sum_{i=1}^{N} \log N(x_i|\mu,\sigma^2) = -N\log\left(\sqrt{2\pi\sigma}\right) - \sum_{i=1}^{N} \frac{(x_i-\mu)^2}{2\sigma^2}$$

$$0 = \frac{d\log P(\mathcal{D}|\theta)}{d\mu} = \sum_{i=1}^{N} \frac{(x_i-\mu)}{\sigma^2} = \frac{\sum_{i=1}^{N} x_i - N\mu}{\sigma^2} \iff$$

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

# ML estimation of Gaussian parameters

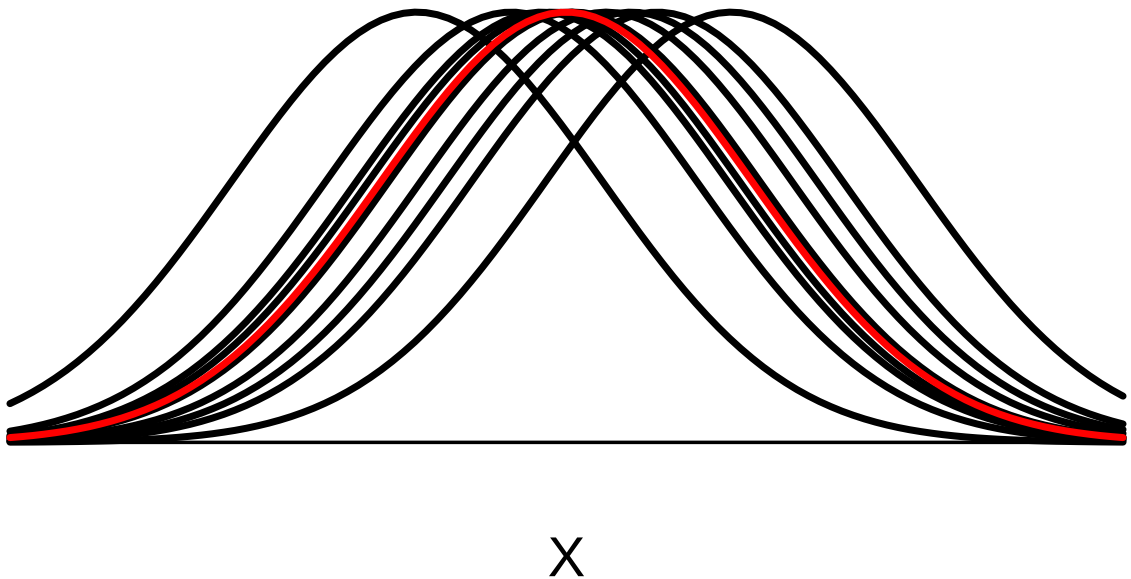
$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

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

- same result by minimizing the sum of square errors!
- but we make assumptions explicit

# Problem: few data points

10 repetitions with 5 points each



# Maximum a Posteriori Estimation

$$\hat{\mu}, \hat{\sigma}^2 = \arg\max_{\mu,\sigma^2} \left[ \prod_{i=1}^{N} P(x_i|\mu, \sigma^2) P(\mu, \sigma^2) \right]$$

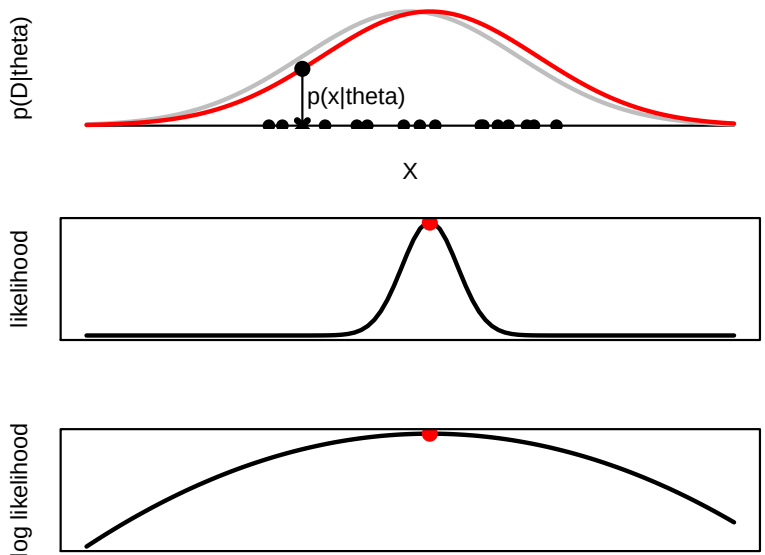
where the prior $P(\mu, \sigma^2)$ needs a nice mathematical form for closed solution

$$\begin{aligned}
\hat{\mu}_{\text{MAP}} &= \frac{N}{N+\gamma} \hat{\mu}_{\text{ML}} + \frac{\gamma}{N+\gamma} \delta \\
\hat{\sigma}^2_{\text{MAP}} &= \frac{N}{N+3+2\alpha} \hat{\sigma}^2_{\text{ML}} + \frac{2\beta + \gamma(\delta + \hat{\mu}_{\text{MAP}})^2}{N+3+2\alpha}
\end{aligned}$$

where $\alpha, \beta, \gamma, \delta$ are parameters of the prior distribution

# ML, MAP and Point Estimates

- Both ML and MAP produce point estimates of $\theta$
- Assumption: there is a true value for $\theta$
- advantage: once $\hat{\theta}$ is found, everything is known



# Overfitting

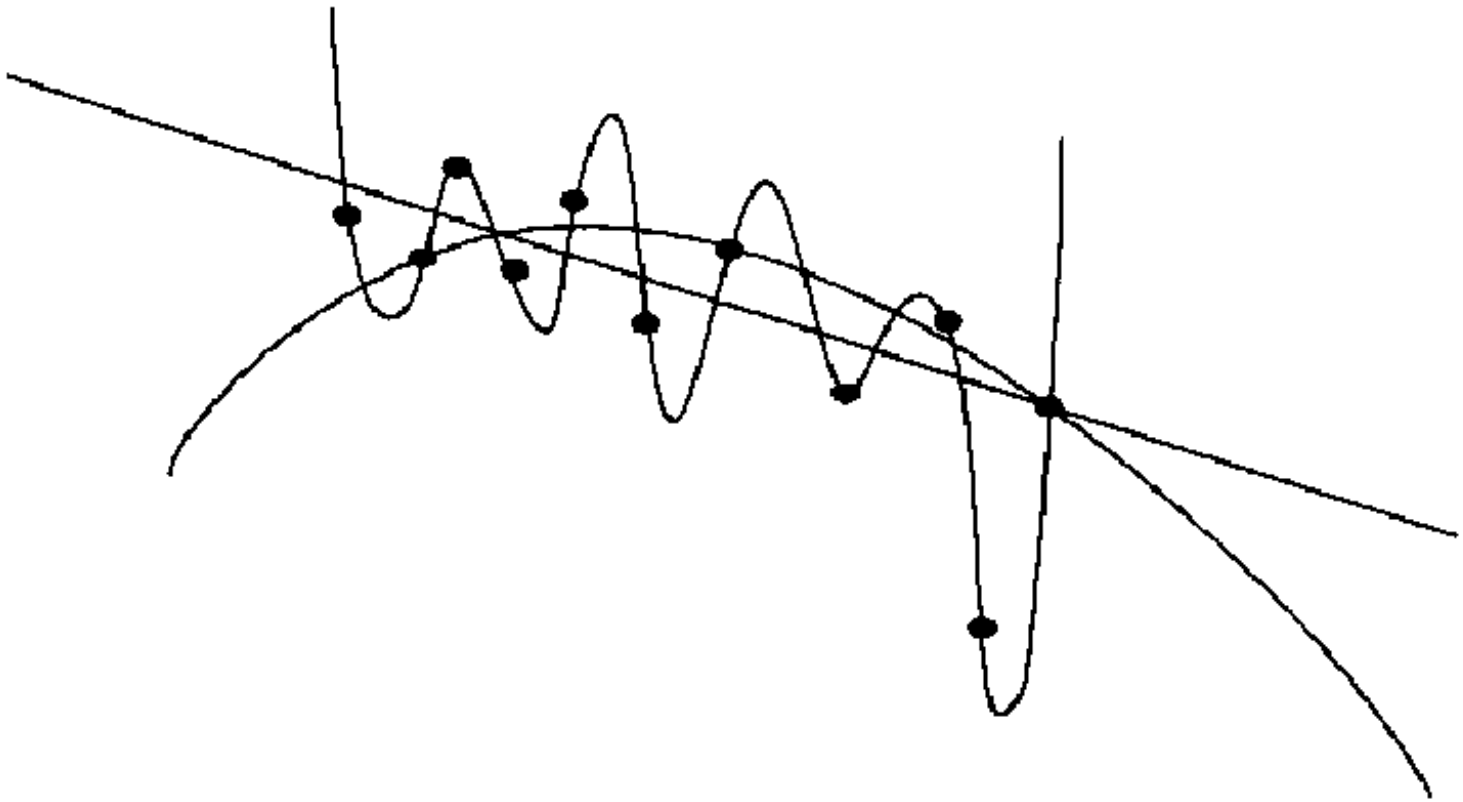


Figure from Huang, Acero, Hon.

# Overfitting: Phoneme Discrimination

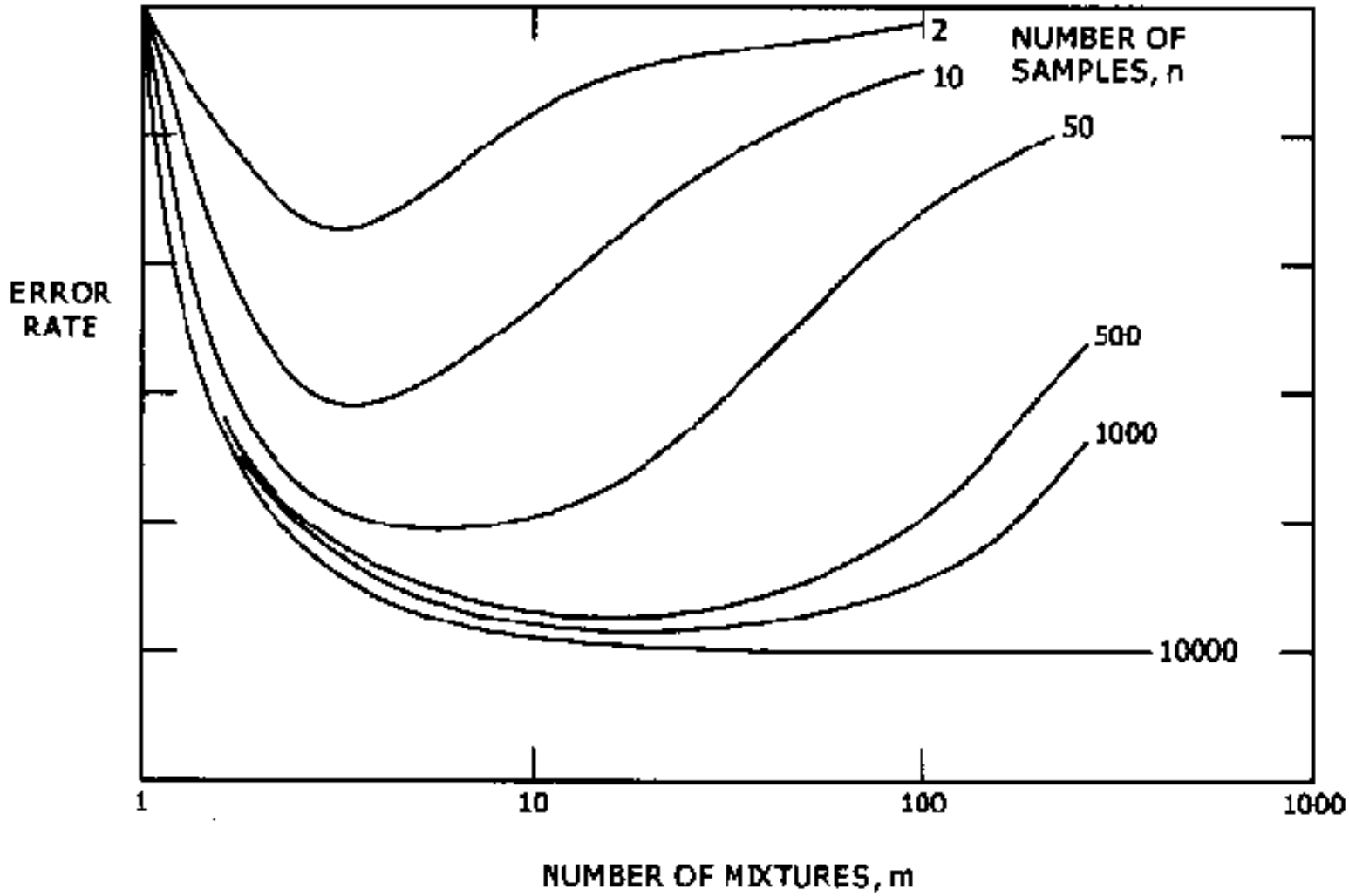



Figure from Huang, Acero, Hon.

# Bayesian estimation

- Consider $\theta$ as a random variable
- characterize $\theta$ with the posterior distribution $P(\theta|\mathcal{D})$ given the data

$$\begin{array}{lll} \text{ML:} & \mathcal{D} & \rightarrow \hat{\theta}_{\text{ML}} \\ \text{MAP:} & \mathcal{D}, P(\theta) & \rightarrow \hat{\theta}_{\text{MAP}} \\ \text{Bayes:} & \mathcal{D}, P(\theta) & \rightarrow P(\theta|\mathcal{D}) \end{array}$$

- for new data points, instead of $P(\mathbf{x}_{\text{new}}|\hat{\theta}_{\text{ML}})$ or $P(\mathbf{x}_{\text{new}}|\hat{\theta}_{\text{MAP}})$, compute:

$$P(\mathbf{x}_{\text{new}}|\mathcal{D}) = \int_{\theta \in \Theta} P(\mathbf{x}_{\text{new}}|\theta) P(\theta|\mathcal{D}) d\theta$$

# Bayesian estimation (cont.)

Pros:

- better use of the data
- makes a priori assumptions explicit
- easily implemented recursively
  - use posterior $p(\theta|\mathcal{D})$ as new prior
- reduce overfitting

Cons:

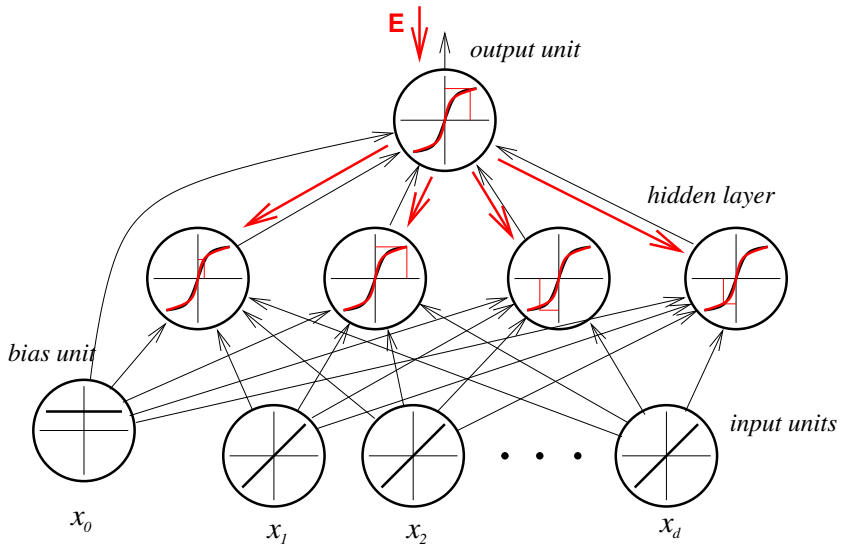
- definition of noninformative priors can be tricky
- often requires numerical integration

# Other Training Strategies: Discriminative Training

- Maximum Mutual Information Estimation
- Minimum Error Rate Estimation
- Neural Networks

# Multi layer neural networks

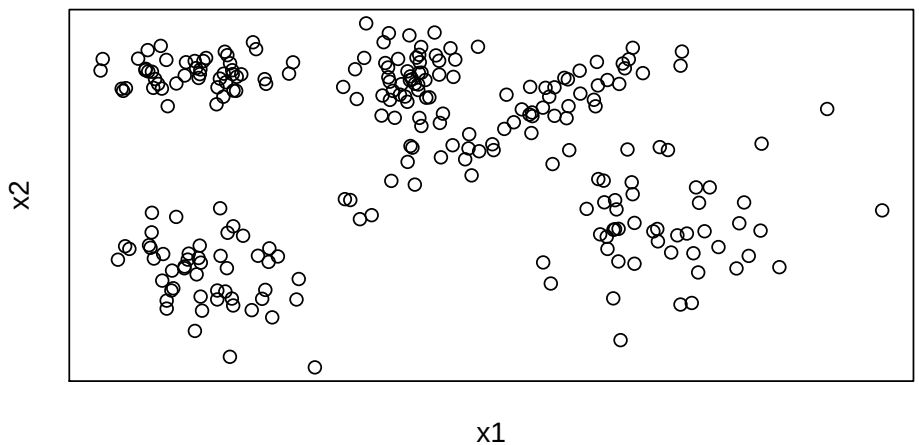
Multi layer
neural networks




- Backpropagation algorithm

# Unsupervised Learning

- so far we assumed we knew the class $\omega_i$ for each data point
- what if we don't?
- class independence assumption loses meaning



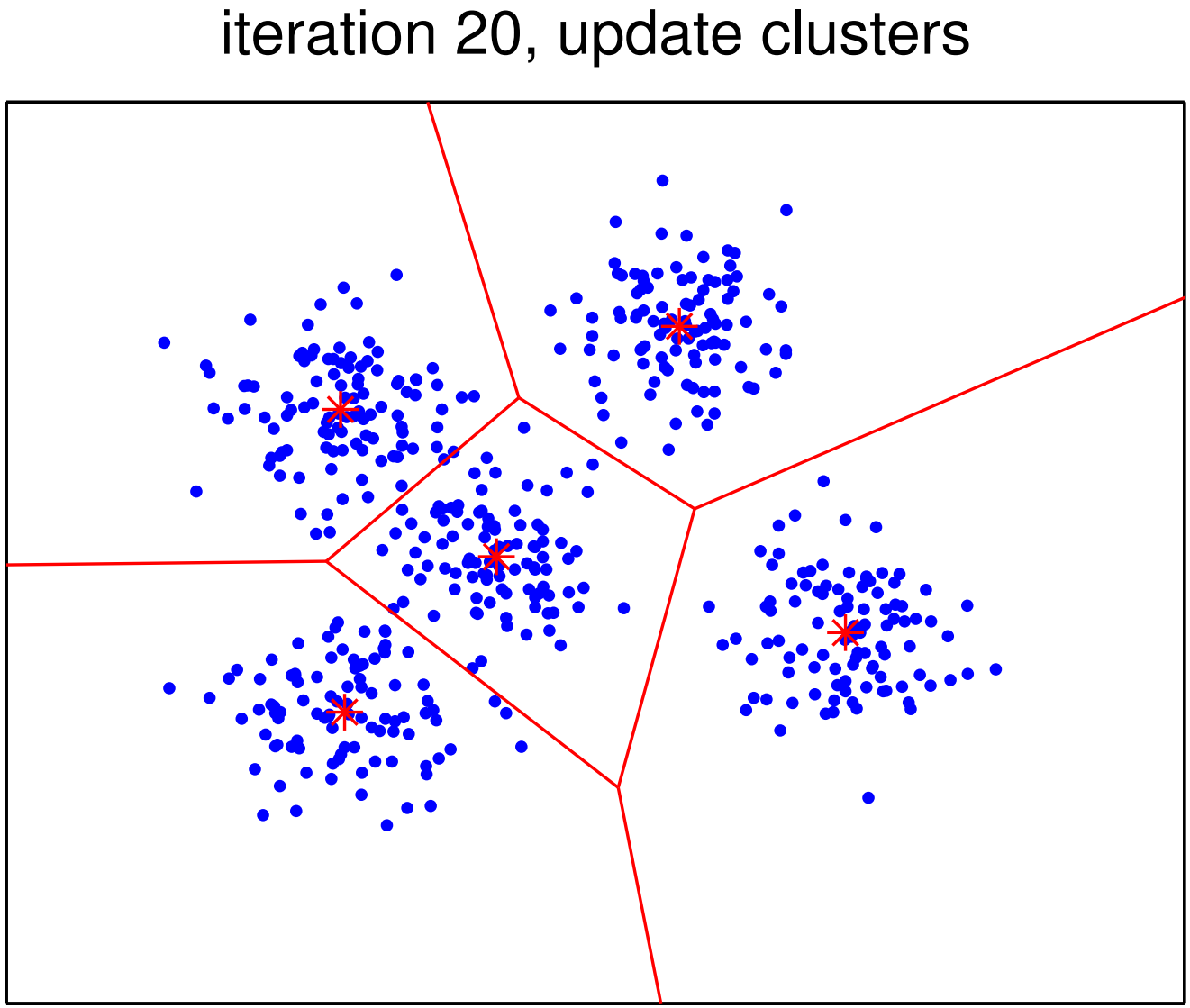
# Vector Quantisation, K-Means

- describes each class with a centroid
- a point belongs to a class if the corresponding centroid is closest (Euclidean distance)
- iterative procedure
- guaranteed to converge
- not guaranteed to find the optimal solution
- used in vector quantization

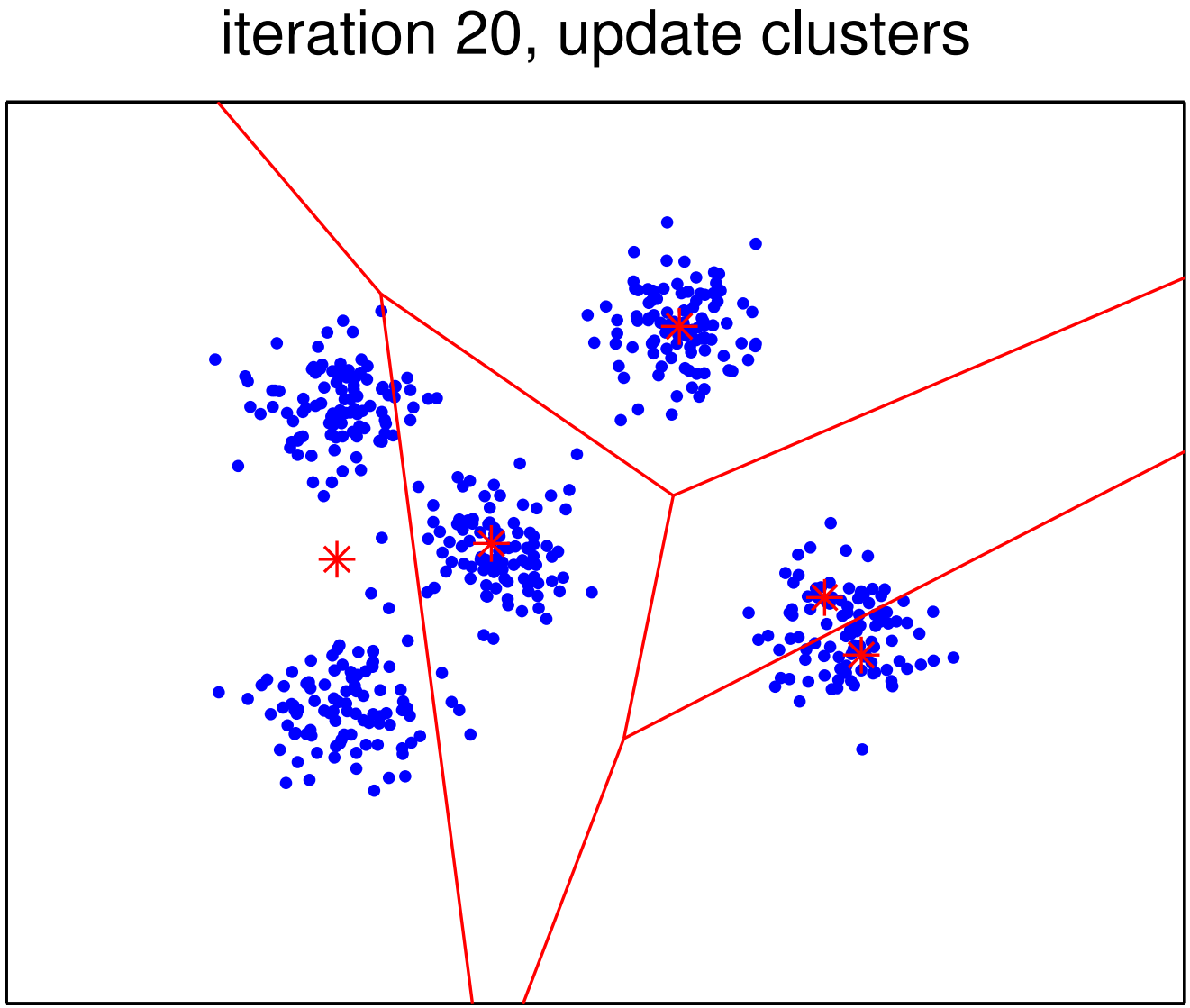
# K-means: algorithm

**Data:** $k$ (number of desired clusters), $n$ data points $\mathbf{x}_i$
**Result:** $k$ clusters
initialization: assign initial value to $k$ centroids $\mathbf{c}_i$;
**repeat**
    assign each point $\mathbf{x}_i$ to closest centroid $\mathbf{c}_j$;
    compute new centroids as mean of each group of points;
**until** *centroids do not change*;
**return** $k$ clusters;

# K-means: example



# K-means: sensitivity to initial conditions



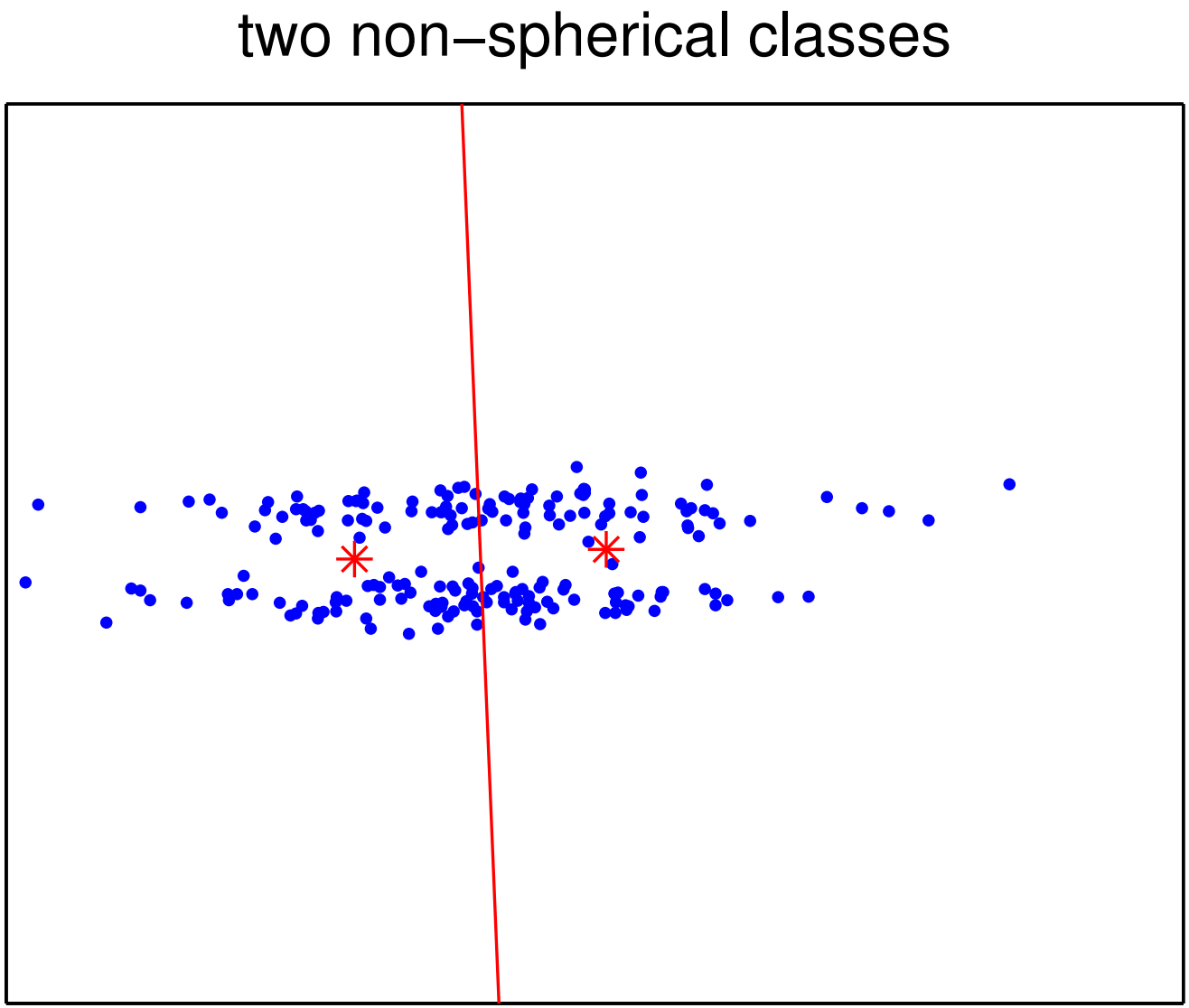
# Solution: LBG Algorithm

- Linde–Buzo–Gray
- start with one centroid
- adjust to mean
- split centroid (with $\epsilon$)
- K-means
- split again...

# K-means: limits of Euclidean distance

- the Euclidean distance is isotropic (same in all directions in $\mathbb{R}^p$)
- this favours spherical clusters
- the size of the clusters is controlled by their distance

# K-means: non-spherical classes



# Probabilistic Clustering

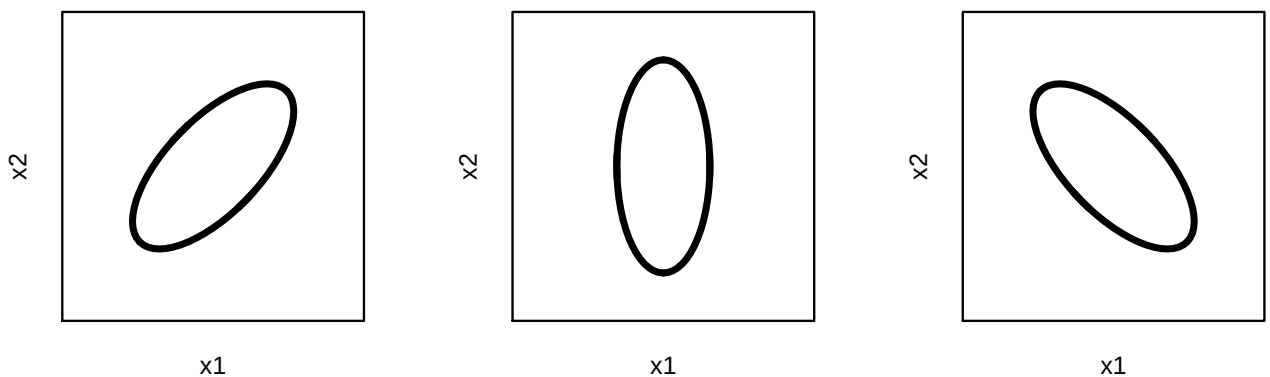
- model data as a mixture of probability distributions (Gaussian)
- each distribution corresponds to a cluster
- clustering corresponds to parameter estimation

# Gaussian distributions

$$f_k(\mathbf{x}_i|\mu_k, \Sigma_k) = \frac{exp\left\{-\frac{1}{2}(\mathbf{x}_i - \mu_k)^T \Sigma_k^{-1}(\mathbf{x}_i - \mu_k)\right\}}{(2\pi)^{\frac{p}{2}} |\Sigma_k|^{\frac{1}{2}}}$$

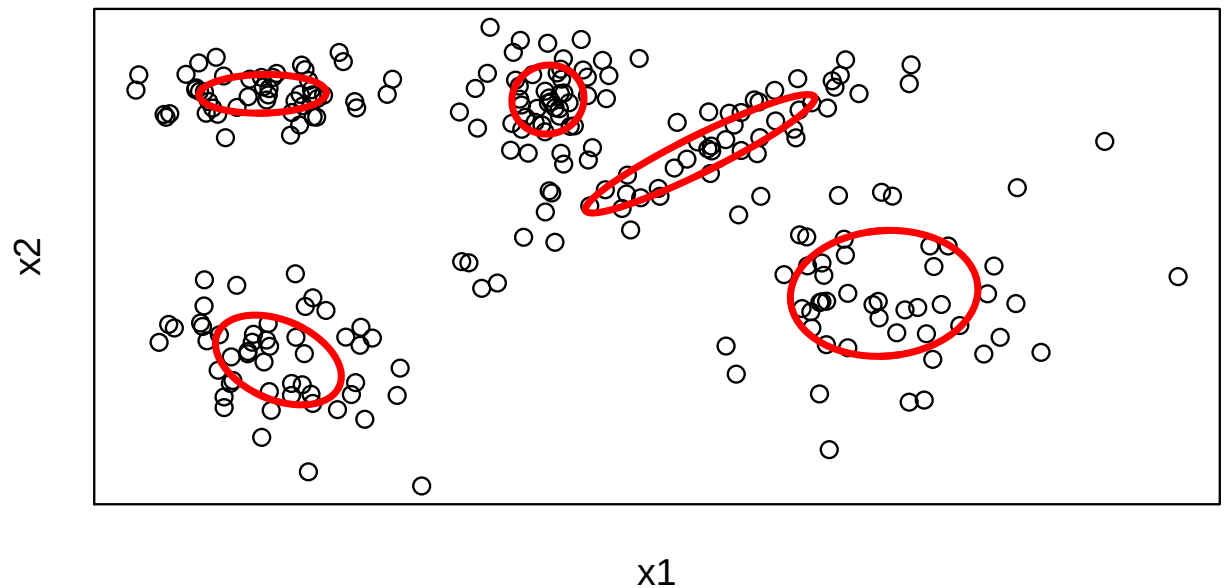
Eigenvalue decomposition of the covariance matrix:

$$\Sigma_k = \lambda_k D_k A_k D_k^T$$



# Mixture of Gaussian distributions

| $\Sigma_k$ | Distribution | Volume | Shape | Orientation |
|---|---|---|---|---|
| $\lambda I$ | Spherical | Equal | Equal | N/A |
| $\lambda_k I$ | Spherical | Variable | Equal | N/A |
| $\lambda DAD^T$ | Ellipsoidal | Equal | Equal | Equal |
| $\lambda D_k A D_k^T$ | Ellipsoidal | Equal | Equal | Variable |
| $\lambda_k D_k A D_k^T$ | Ellipsoidal | Variable | Equal | Variable |
| $\lambda_k D_k A_k D_k^T$ | Ellipsoidal | Variable | Variable | Variable |



# Fitting the model

- given the data $D = \{\mathbf{x}_i\}$
- given a certain model $\mathcal{M}$ and its parameters $\theta$
- maximize the model fit to the data as expressed by the likelihood

$$\mathcal{L} = p(D|\theta)$$

# Unsupervised Case

- release class independence assumption:
- learn the mixture at once
- problem of missing data
- solution: Expectation Maximization

# Expectation Maximization

Fitting model parameters with missing (latent) variables

$$P(\mathbf{x}|\theta) = \sum_{k=1}^{K} \pi_k P(x|\theta_k),$$

$$\text{with } \theta = \{\pi_1, \ldots, \pi_k, \theta_1, \ldots, \theta_K\}$$

- very general idea (applies to many different probabilistic models)
- augment the data with the missing variables: $h_{ik}$ probability of assignment of each data point $x_i$ to each component of the mixture $k$
- optimize the Likelihood of the complete data:

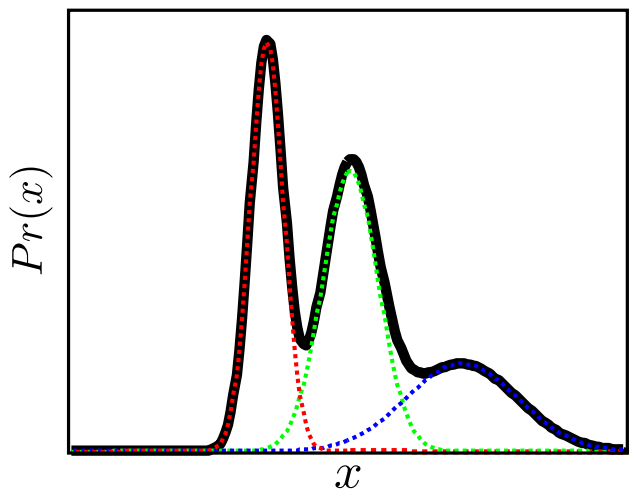
$$P(\mathbf{x}, \mathbf{h}|\theta)$$

# Mixture of Gaussians

This distribution is a weight sum of $K$ Gaussian distributions

$$P(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \sigma_k^2)$$

where $\pi_1 + \cdots + \pi_K = 1$
and $\pi_k > 0$ $(k = 1, \ldots, K)$.




This model can describe **complex multi-modal** probability distributions by combining simpler distributions.

# Mixture of Gaussians

$$P(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \sigma_k^2)$$

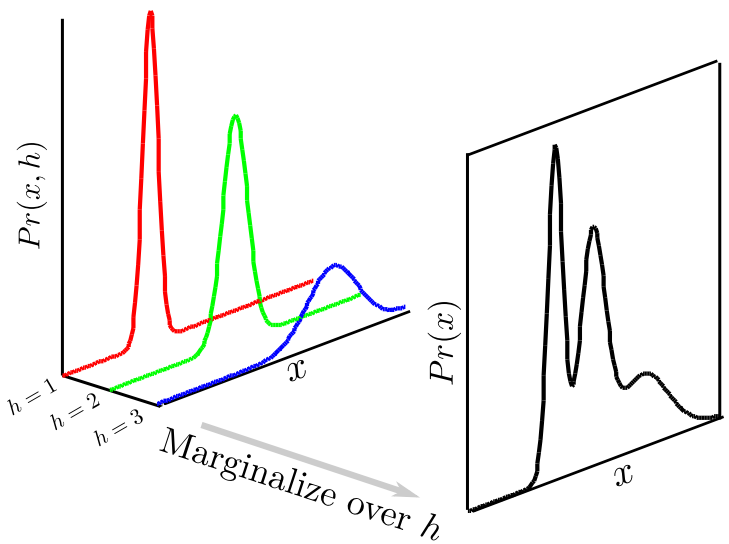
- Learning the parameters of this model from training data $x_1, \ldots, x_n$ is not trivial - using the usual straightforward maximum likelihood approach.
- Instead learn parameters using the **Expectation-Maximization** (EM) algorithm.

# Mixture of Gaussians as a marginalization

We can interpret the Mixture of Gaussians model with the introduction of a discrete hidden/latent variable $h$ and $P(x, h)$:

$$P(x) = \sum_{k=1}^{K} P(x, h = k) = \sum_{k=1}^{K} P(x \mid h = k) P(h = k)$$

$$= \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \sigma_k^2)$$




← **mixture density**

Figures taken from **Computer Vision: models, learning and inference** by Simon Prince.

# EM for two Gaussians

**Assume:** We know the pdf of $x$ has this form:

$$P(x) = \pi_1 \mathcal{N}(x; \mu_1, \sigma_1^2) + \pi_2 \mathcal{N}(x; \mu_2, \sigma_2^2)$$

where $\pi_1 + \pi_2 = 1$ and $\pi_k > 0$ for components $k = 1, 2$.

**Unknown:** Values of the parameters (Many!)

$$\Theta = (\pi_1, \mu_1, \sigma_1, \mu_2, \sigma_2).$$

**Have:** Observed $n$ samples $x_1, \ldots, x_n$ drawn from $P(x)$.

**Want to:** Estimate $\Theta$ from $x_1, \ldots, x_n$.

**How would it be possible to get them all???**

# EM for two Gaussians

For each sample $x_i$ introduce a *hidden variable* $h_i$

$$h_i = \begin{cases} 1 & \text{if sample } x_i \text{ was drawn from } \mathcal{N}(x; \mu_1, \sigma_1^2) \\ 2 & \text{if sample } x_i \text{ was drawn from } \mathcal{N}(x; \mu_2, \sigma_2^2) \end{cases}$$

and come up with initial values

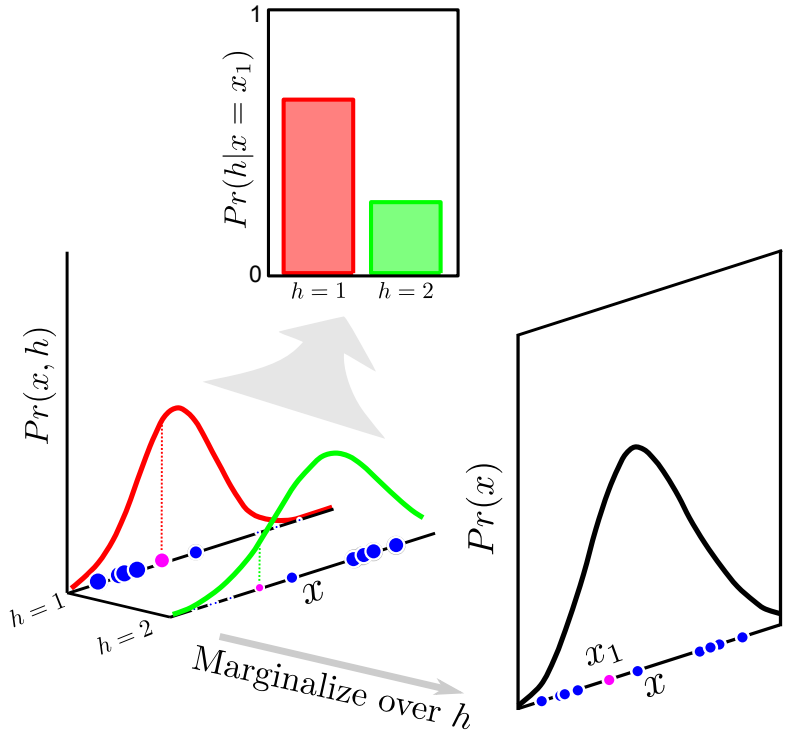
$$\Theta^{(0)} = (\pi_1^{(0)}, \mu_1^{(0)}, \sigma_1^{(0)}, \mu_2^{(0)}, \sigma_2^{(0)})$$

for each of the parameters.

EM is an *iterative algorithm* which updates $\Theta^{(t)}$ using the following two steps...

# EM for two Gaussians: E-step

The responsibility of $k$-th Gaussian for each sample $x$ (indicated by the size of the projected data point)




**Look at each sample $x$ along hidden variable $h$ in the E-step**

Figure from **Computer Vision: models, learning and inference** by Simon Prince.

# EM for two Gaussians: E-step (cont.)

**E-step:** Compute the *"posterior probability"* that $x_i$ was generated by component $k$ given the current estimate of the parameters $\Theta^{(t)}$. (responsibilities)

for $i = 1, \ldots n$

for $k = 1, 2$

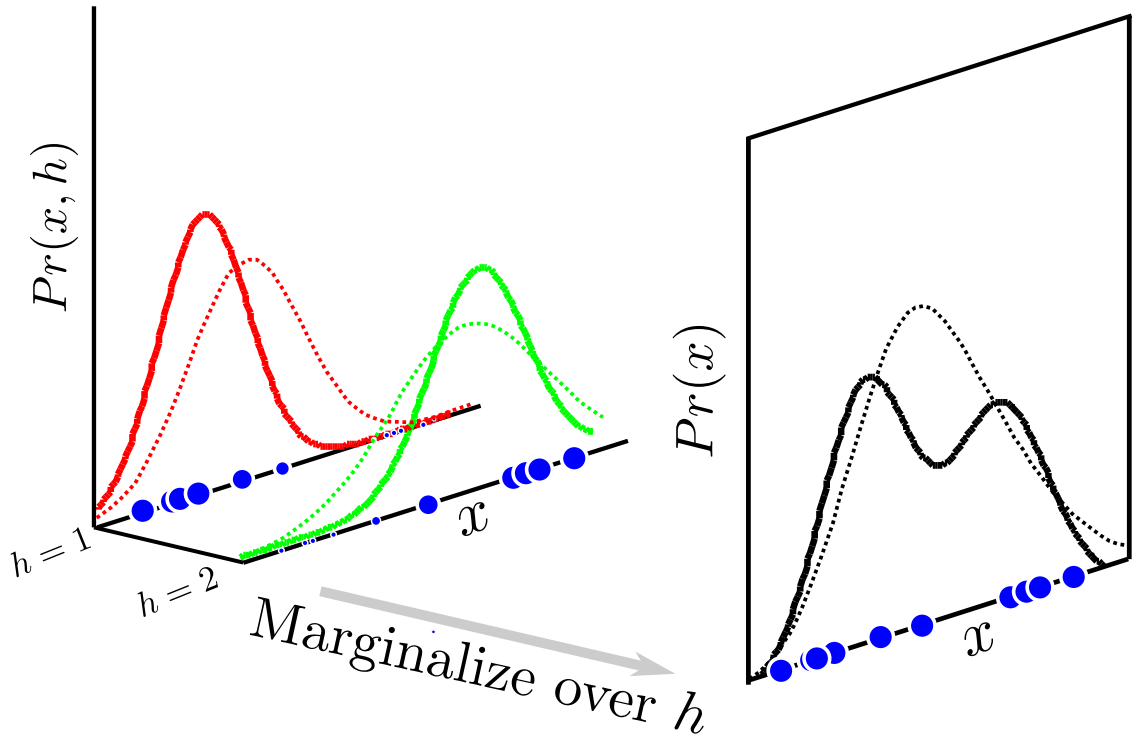
$$\gamma_{ik}^{(t)} = P(h_i = k \mid x_i, \Theta^{(t)})$$

$$= \frac{\pi_k^{(t)} \mathcal{N}(x_i; \mu_k^{(t)}, \sigma_k^{(t)})}{\pi_1^{(t)} \mathcal{N}(x_i; \mu_1^{(t)}, \sigma_1^{(t)}) + \pi_2^{(t)} \mathcal{N}(x_i; \mu_2^{(t)}, \sigma_2^{(t)})}$$

**Note:** $\gamma_{i1}^{(t)} + \gamma_{i2}^{(t)} = 1$ and $\pi_1 + \pi_2 = 1$

# EM for two Gaussians: M-step

Fitting the Gaussian model for each of $k$-th constinuetnt.
Sample $x_i$ contributes according to the responsibility $\gamma_{ik}$.




(dashed and solid lines for fit before and after update)

**Look along samples $x$ for each $h$ in the M-step**

Figure from **Computer Vision: models, learning and inference** by Simon Prince.

# EM for two Gaussians: M-step (cont.)

**M-step:** Compute the *Maximum Likelihood* of the parameters of the mixture model given out data's membership distribution, the $\gamma_i^{(t)}$'s:

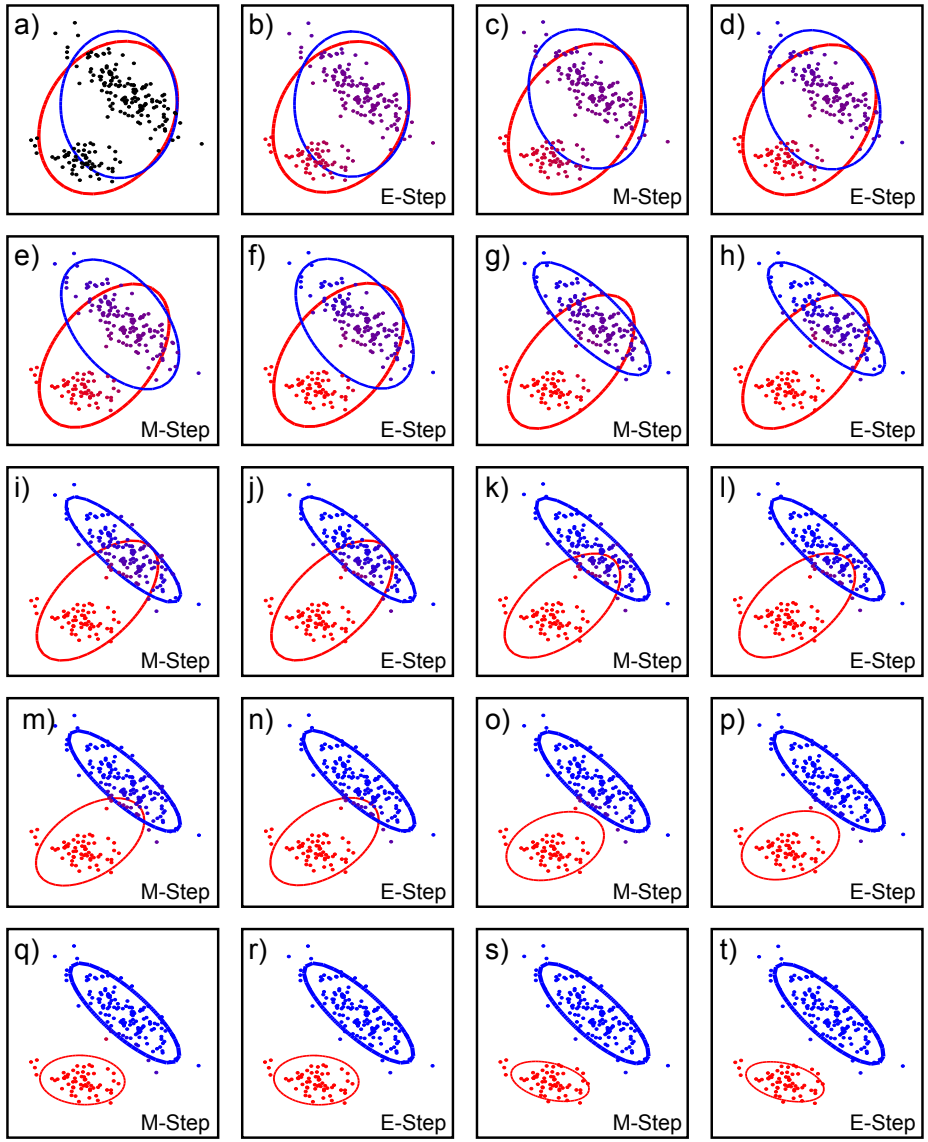
for $k = 1, 2$

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{ik}^{(t)} x_i}{\sum_{i=1}^{n} \gamma_{ik}^{(t)}},$$

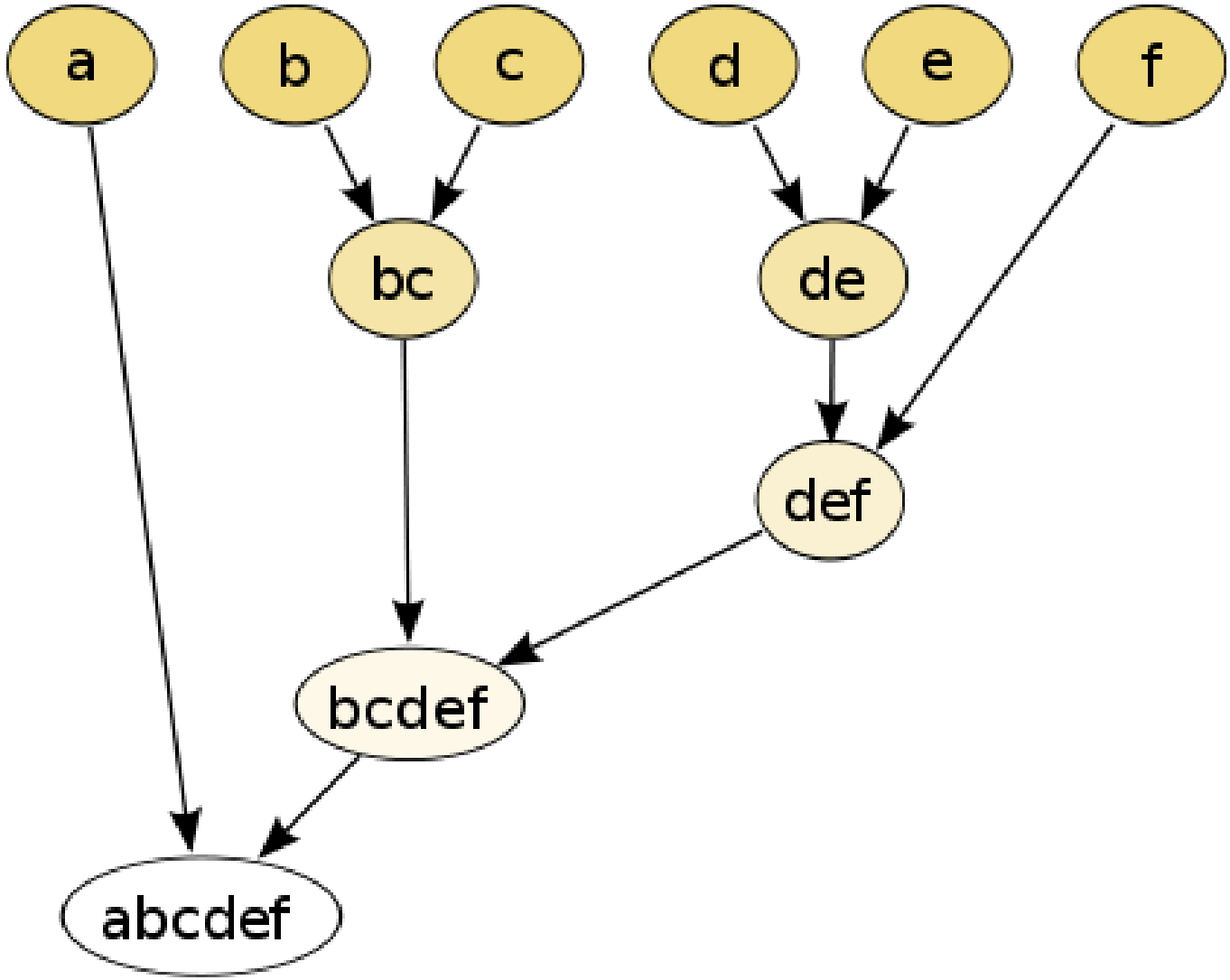
$$\sigma_k^{(t+1)} = \sqrt{\frac{\sum_{i=1}^{n} \gamma_{ik}^{(t)} \left(x_i - \mu_k^{(t+1)}\right)^2}{\sum_{i=1}^{n} \gamma_{ik}^{(t)}}},$$

$$\pi_k^{(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{ik}^{(t)}}{n}.$$

# EM in practice



# Hierarchical Clustering




(Figure from Wikipedia)