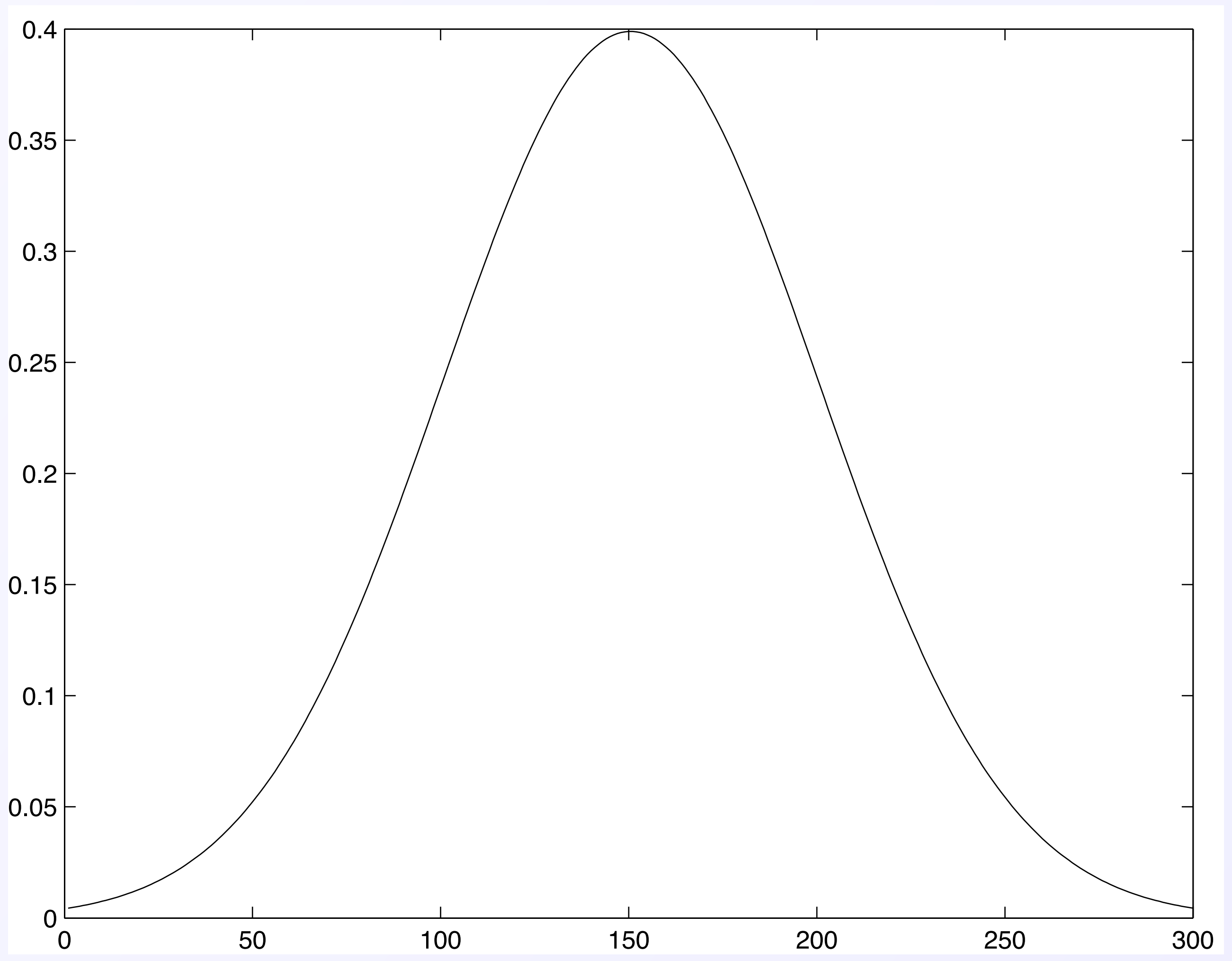
L4
From distances to distributions

COM4511/COM6511 - Speech Technology



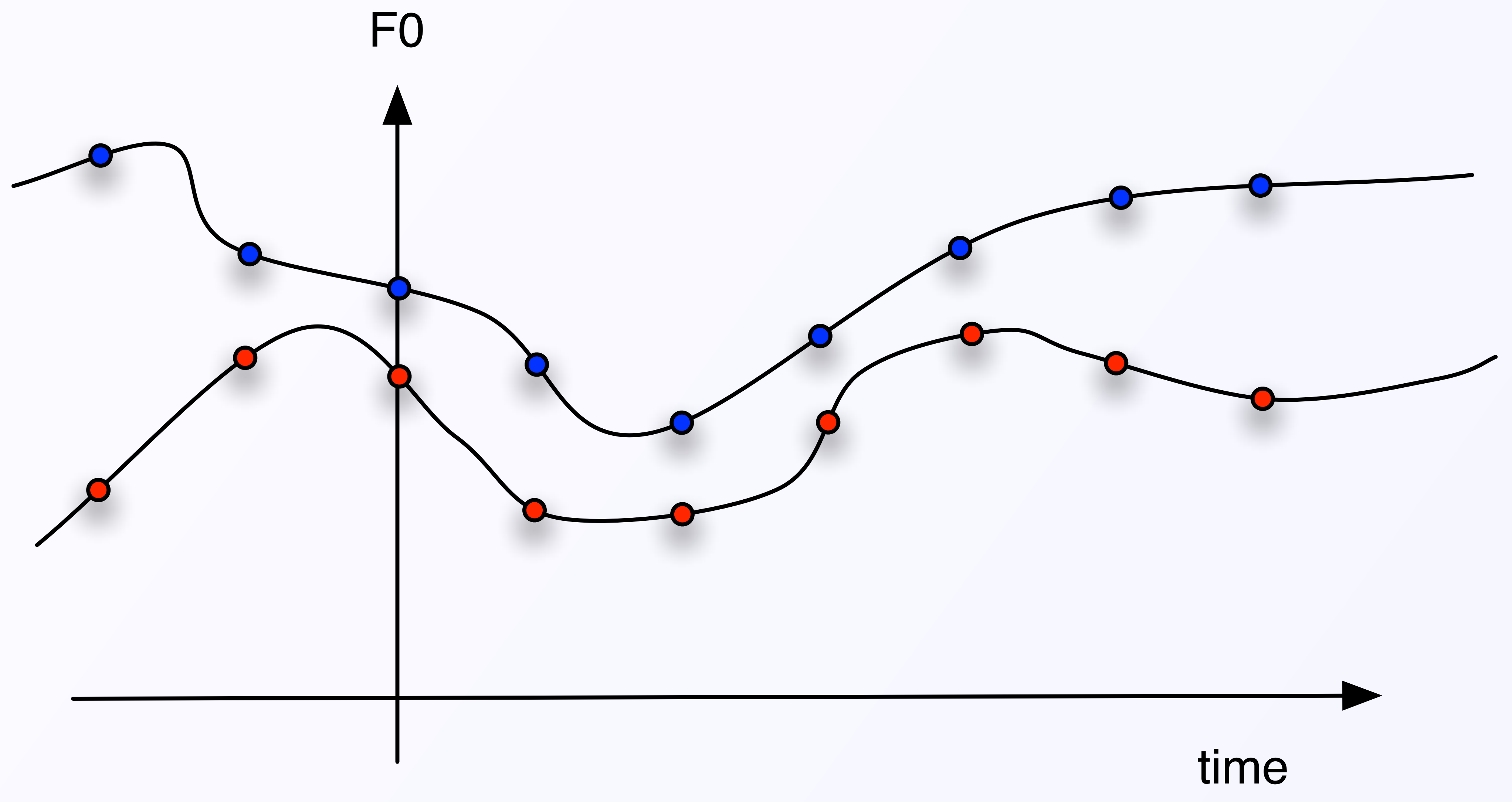
Thomas Hain
t.hain@sheffield.ac.uk
Spring Semester

Outline

- Feature strings
- K means clustering
- Gaussian models
- Maximum likelihood

Recognising speaker maturity

- **1D Features**
  Pitch contour from young or adult speakers
  - F0 is measurable equivalent of pitch
  - In reality F0 measurements are not that nice !



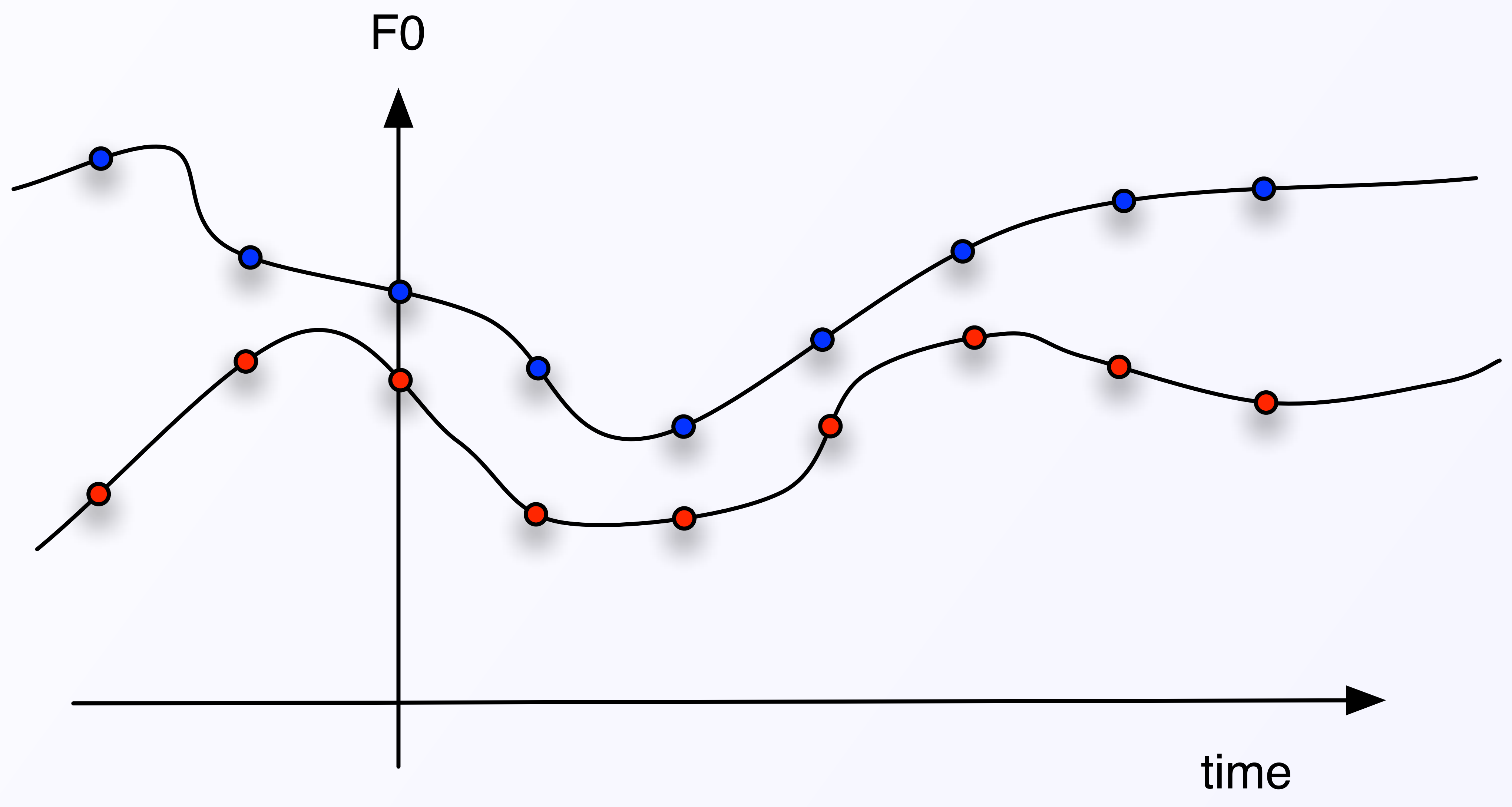
-2	-1	0	1	2	3	4	5
210	190	180	150	130	160	180	190
110	170	160	120	120	150	190	180

How do we decide if the speech is from a child or an adult speaker ?

Value by value decision

- Compare against a threshold

$$pitch(t) \lessgtr \theta$$



- How do we intuitively set the threshold ?
  - We need data
  - 2 (good) options !

What are the problems ?

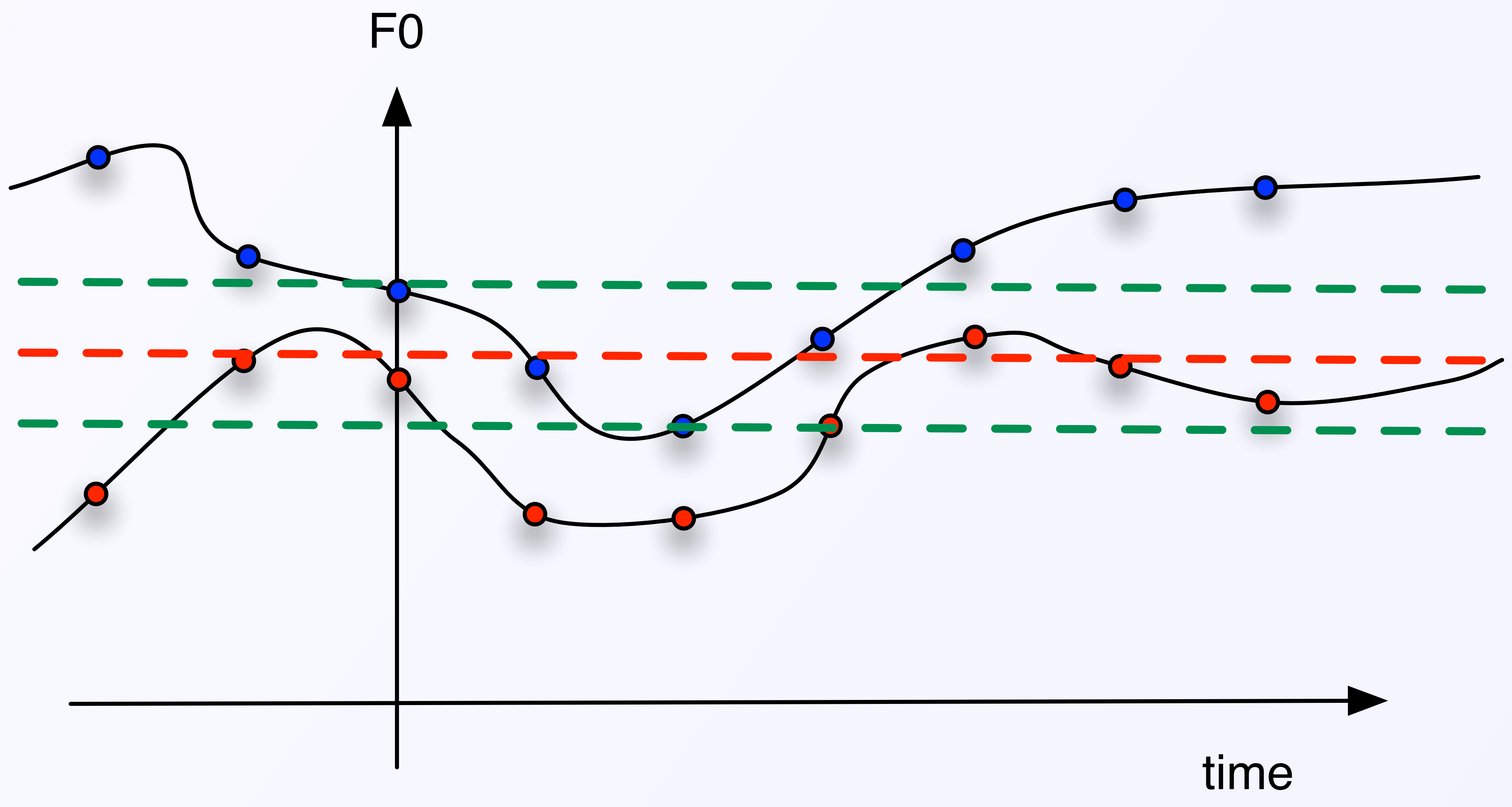
Value by value decision

- Learning the parameter θ

$$\theta_c = \frac{1}{T}\sum_{t=1}^{T} pitch_c(t) \qquad \theta_a = \frac{1}{T}\sum_{t=1}^{T} pitch_a(t)$$

- put the threshold in the middle

$$\theta = \frac{\theta_c + \theta_a}{2}$$



What are the problems ?

Learning from data - formal concepts

- **Cost function**
  - Common sense: The further each value is away from the threshold the safer was our decision
  - What does *far away* mean ? Need a distance metric:

$$d(x,y)$$

- The overall cost *D* (give a simple threshold model) is

$$D = \sum_{childrensamples} d(pitch(t)|\theta_c) + \sum_{adultsamples} d(pitch(t)|\theta_a)$$

▶ The goal is to have **overall minimal cost**.

Distance metrics

It often arises that it is necessary to compare a speech vector $\mathbf{o}$ with another speech vector $\mathbf{v}$. The distance measure (or metric) $d(\mathbf{o}, \mathbf{v})$ is defined such that

$$\begin{aligned} d(\mathbf{o}, \mathbf{v}) &= \text{large if } \mathbf{o} \text{ and } \mathbf{v} \text{ are different} \\ &= \text{small if } \mathbf{o} \text{ and } \mathbf{v} \text{ are similar} \end{aligned}$$

More formally, for any three vectors (defining points in space), $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ a distance metric has the properties:

1. $d(\mathbf{x}, \mathbf{x}) = 0$
2. $d(\mathbf{x}, \mathbf{y}) \geq 0$
3. $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$

Simplest is distance metric analogous to physical distance, i.e. Euclidean

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

Other related forms are squared Euclidean (without the square-root above), the maximum distance or the city-block distance.

Distance metrics (2)

The *Mahalanobis distance* is given by

$$d_M(\mathbf{x}, \boldsymbol{\mu}) = (\mathbf{x} - \boldsymbol{\mu})' \, \boldsymbol{\Sigma}^{-1} \, (\mathbf{x} - \boldsymbol{\mu})$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the data. This is the equal (within a constant independent of the vectors) to

- ➜ the negative log of the probability density of vector $\mathbf{x}$ from a multivariate Gaussian of mean $\boldsymbol{\mu}$, covariance $\boldsymbol{\Sigma}$.
- ➜ the Mahalanobis distance is equal to squared the Euclidean distance for the case when $\boldsymbol{\Sigma}$ is the identity matrix.

For the cepstral representations any of the above distance measures are appropriate (since their distributions are close to Gaussian). However for LPC predictor coefficients the Euclidean distance is not appropriate and often the Itakura distance is used.

Itakura distance

The mean squared error (MSE) of a linear predictor is (window length N, predictor order p)

$$E = \sum_{i=1}^{N-1} e^2(n) = \sum_{i=1}^{N-1} (s(n) - \hat{s}(n))^2$$

Using the autocorrelation method, the extended LP coefficient vector $\tilde{\mathbf{a}} = [1 - a_1 - a_2 \cdots - a_p]'$ and the autocorrelation matrix

$$\mathbf{R} = \begin{bmatrix} r_0 & r_1 & \cdots & r_p \\ r_1 & r_0 & \cdots & r_{p-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_p & r_{p-1} & \cdots & r_0 \end{bmatrix}$$

The mean square error of the LP filter on some signal with the autocorrelation matrix $\mathbf{R}$ can be computed by

$$E = \tilde{\mathbf{a}}'\mathbf{R}\tilde{\mathbf{a}}$$

Given two different LP filters with the extended LP coefficient vectors $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ the filters instead of using the Euclidean distance the MSE on some data can be compared. If the difference in MSE is small the filters are considered to be "close". The Itakura distance is defined as

$$d_I = \log\left(\frac{\tilde{\mathbf{a}}'\mathbf{R}\tilde{\mathbf{a}}}{\tilde{\mathbf{b}}'\mathbf{R}\tilde{\mathbf{b}}}\right)$$

Learning from data - deterministic

Taking a (squared) Euclidian distance one obtains the following equation:

$$\begin{aligned} D &= \sum_{t=1}^{T} d(pitch(t), \theta) \\ &= \sum_{t=1}^{T} (pitch(t) - \theta)^2 \end{aligned}$$

- We want to minimise cost !

$$\frac{dD}{d\theta} = 0 \qquad \Rightarrow \theta = ?$$

How does this relate to the result we got before ?

Open issues

1. Our features are normally not 1-dimensional
   - Distances can be computed in higher dimensions BUT we face the ***Curse of Dimensionality***
2. A single value to describe class separation is rather simplistic
   - A better way of describing data is needed
3. Originally we set out to provide probabilities and/or likelihoods, not distances
   - Data likelihood computation is not so different to cost estimation
4. The models do not represent the time-evolving

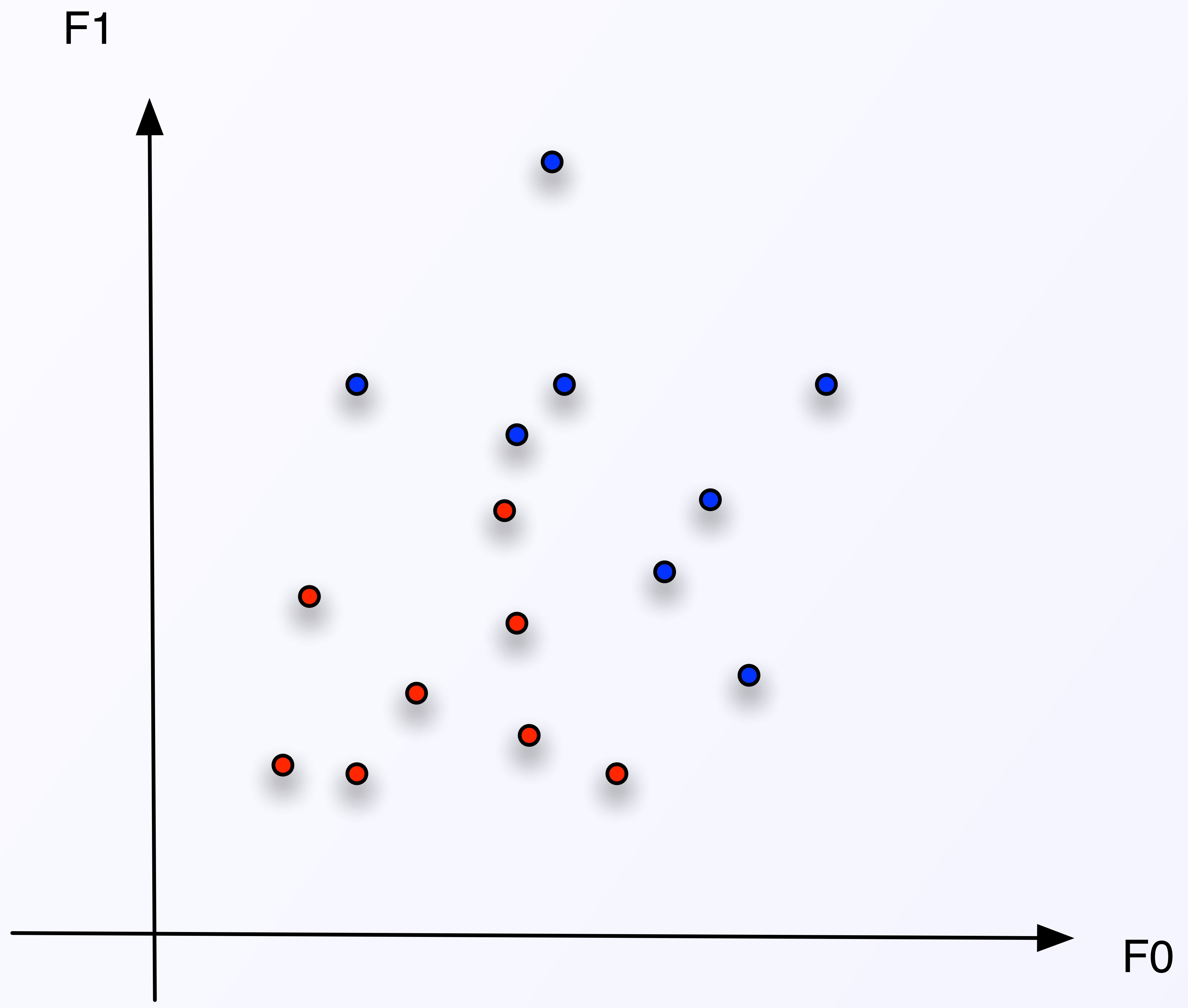
Multi-dimensional data - F0 and F1

- Computing the boundary in the same way as before

$$\theta_c \qquad \theta_a$$

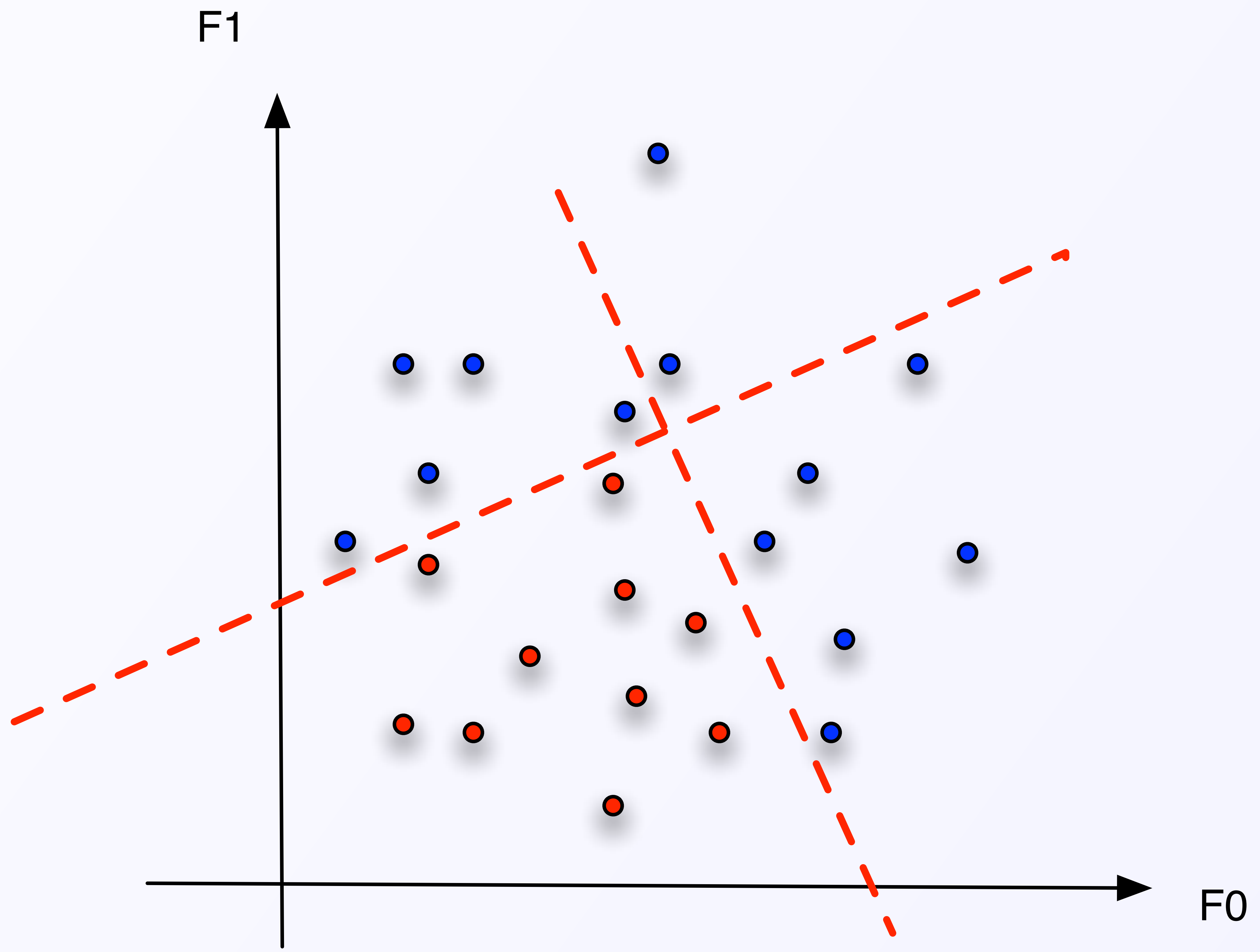
- Here the data is (almost) **linearly separable**

What are the problems ?



Multi-dimensional data

- Data is normally not linearly separable
  - Need several lines to describe boundaries
- **Two options to improve modelling**
  - Describe the boundaries
  - <u>Describe the data</u>
- Before boundaries were found **implicitly** through computing averages of data points



Solution: Vector Quantisation

Vector quantisation

There are a number of instances where it is required to represent a continuously valued multi-dimensional speech vector by a discrete symbol. This process is called **Vector Quantisation** (VQ).

Thus a speech vector $\mathbf{o}$ is mapped to a scalar:

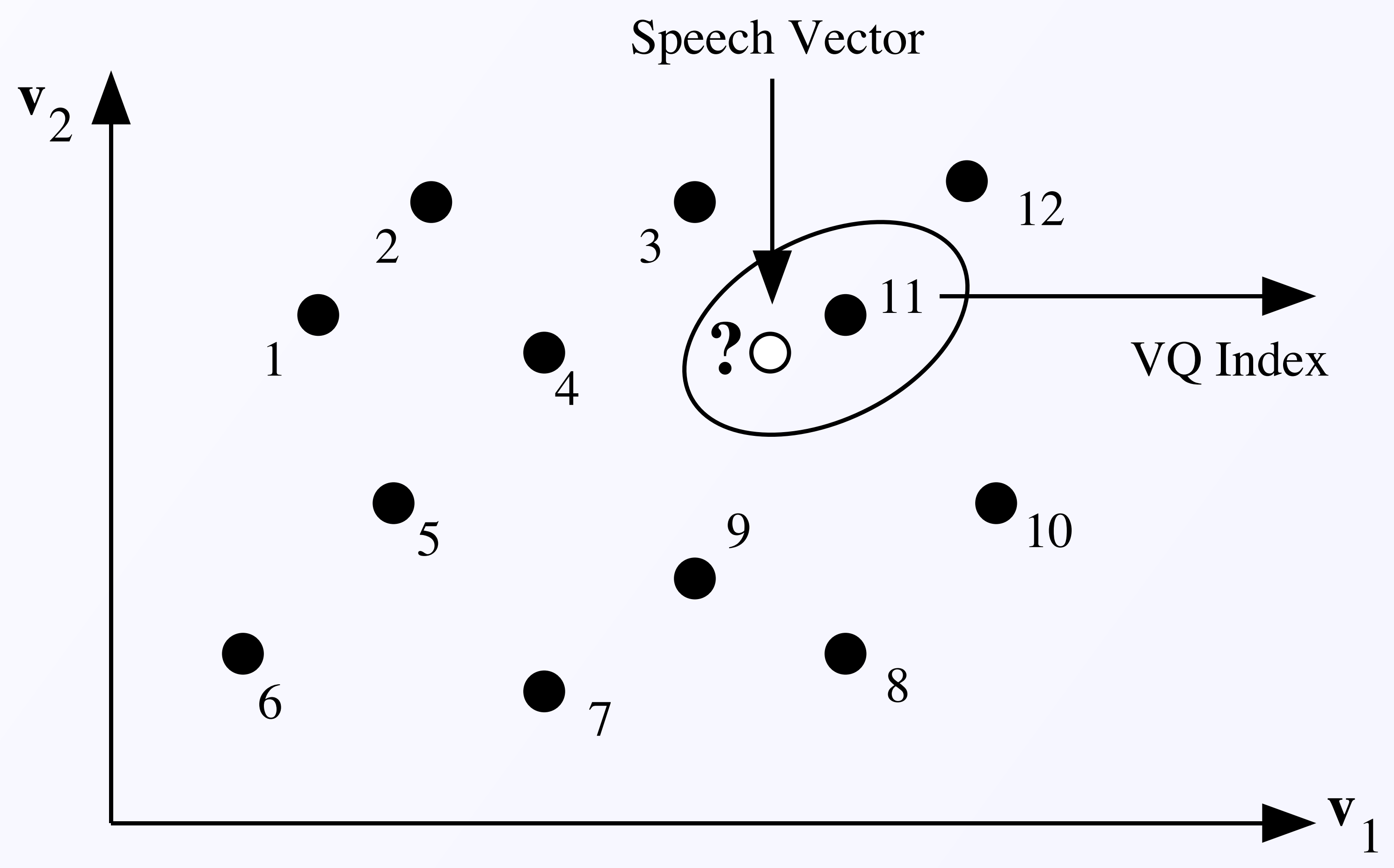
$$\mathbf{o} \quad \rightarrow \quad k$$

The codebook $\mathcal{C}$ consists of a set of K centroid vectors (*codewords*)

$$\mathcal{C} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K\}$$

The observation vectors are then mapped to a codebook symbol using

$$k = \arg \min_{1 \leq l \leq K} d(\mathbf{o}, \mathbf{v}_l)$$



The set of codewords together with a distance metric provide partitioning of the space.

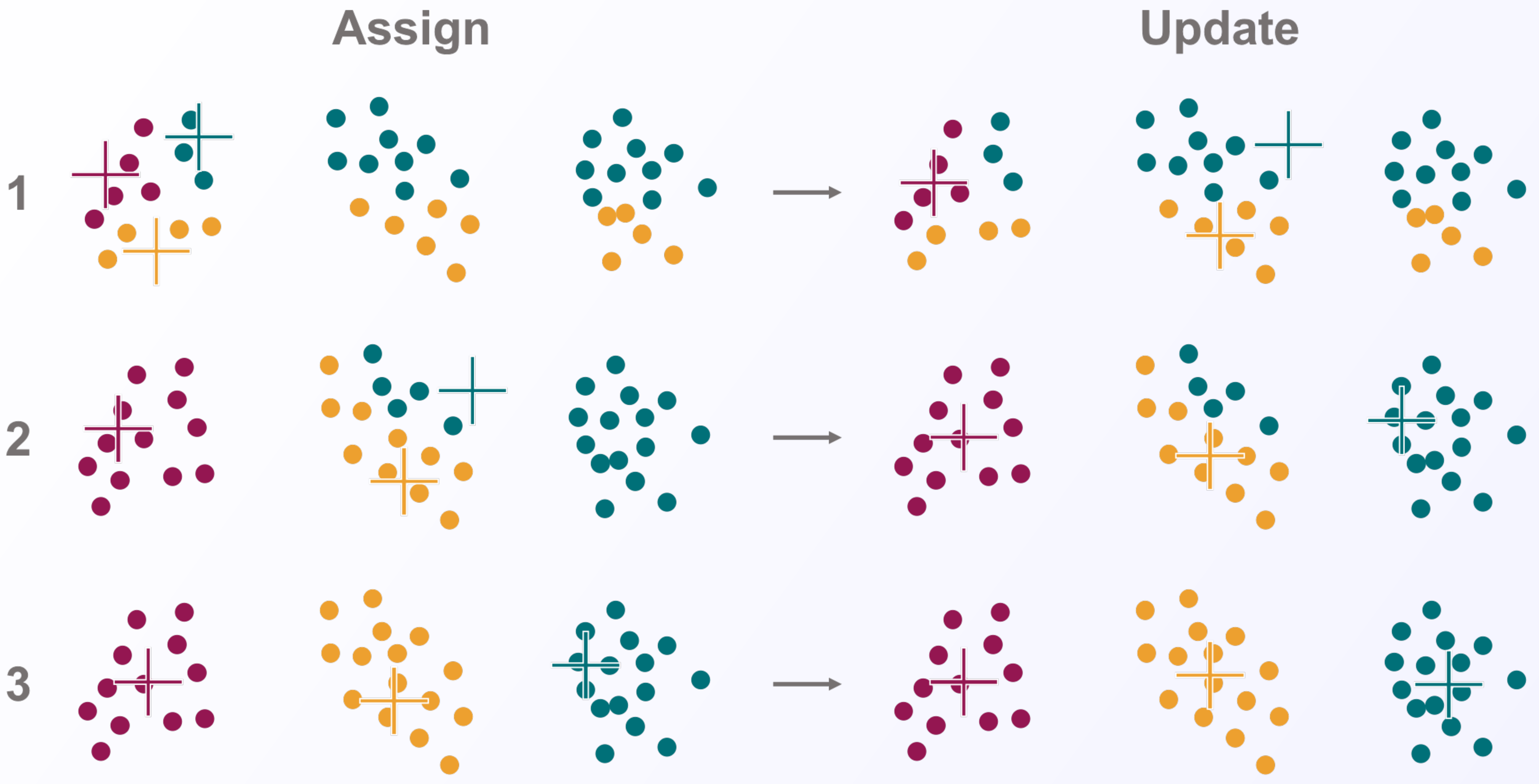
The Linde Buzo Gray (LBG) algorithm

Other names are the *Generalised Lloyd algorithm* or k-means clustering. Variants of k-means clustering are commonly used to train VQ codebooks.

- *Initialisation:* Choose an arbitrary set of K codebook vectors, $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_K\}$
- *Recursion:*
  1. For each speech frame $\mathbf{o}_t$ in the training data calculate codeword $\mathbf{q}$ (classification)
  $$k_t = \arg \min_{1 \leqslant l \leqslant K} \{d(\mathbf{o}_t, \mathbf{v}_l)\}$$
     where $d(\mathbf{o}_t, \mathbf{v}_l)$ is some distortion (distance) measure.
  2. Compute the total distortion that has occurred due to this quantisation.
  $$D = \sum_t d\left(\mathbf{o}_t, \mathbf{v}_{k_t}\right)$$
     If D is sufficiently small **stop.**
  3. Recompute the codebook elements from the assigned observations (sample means).
  4. Return to step 1.

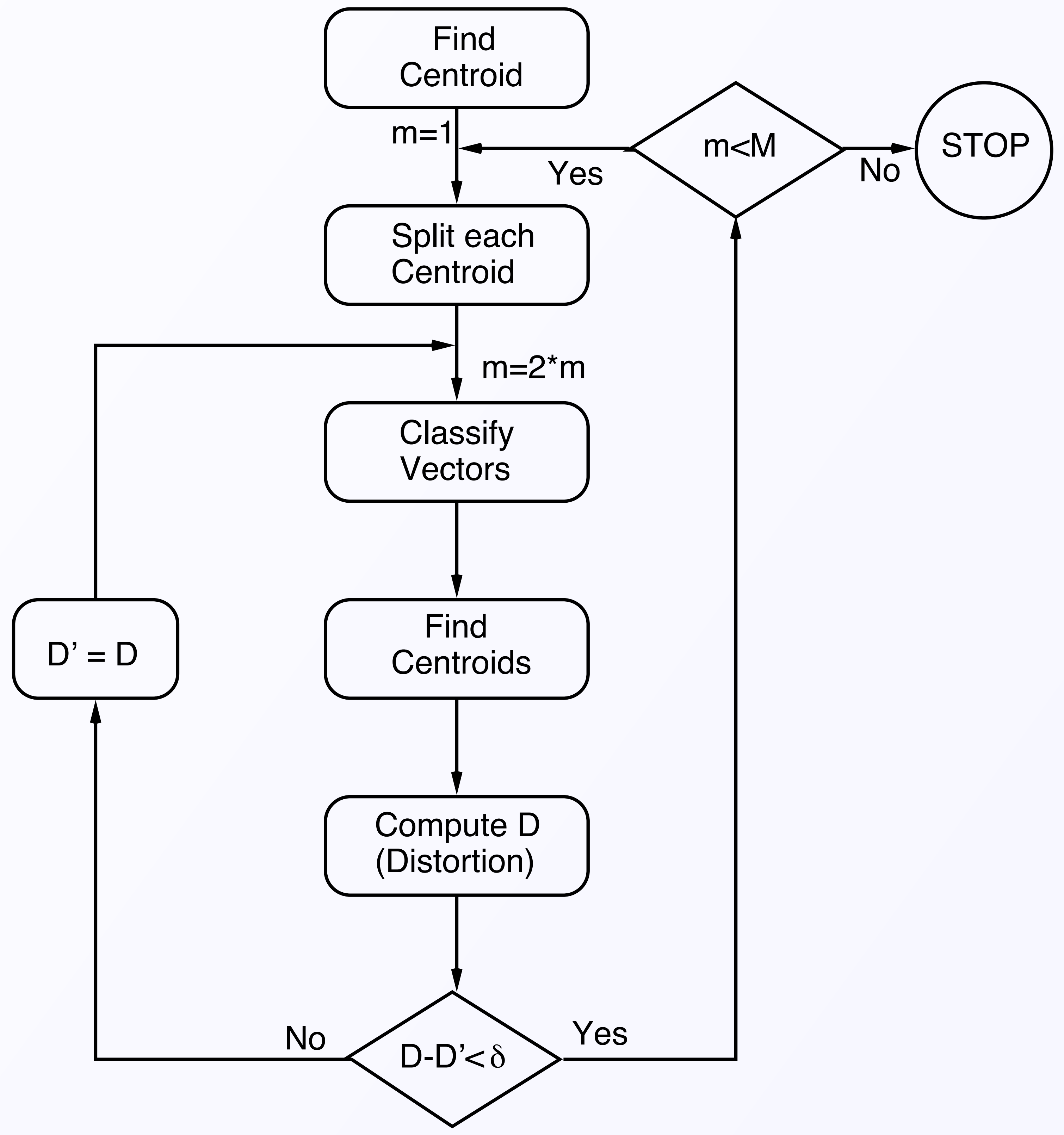
The clustering procedure may produce "empty" means. The algorithm has been shown to converge to a local maximum (take best of multiple runs).

Example - k means



Centroid splitting

There are alternative algorithms which do not require to start from a predefined number of clusters.

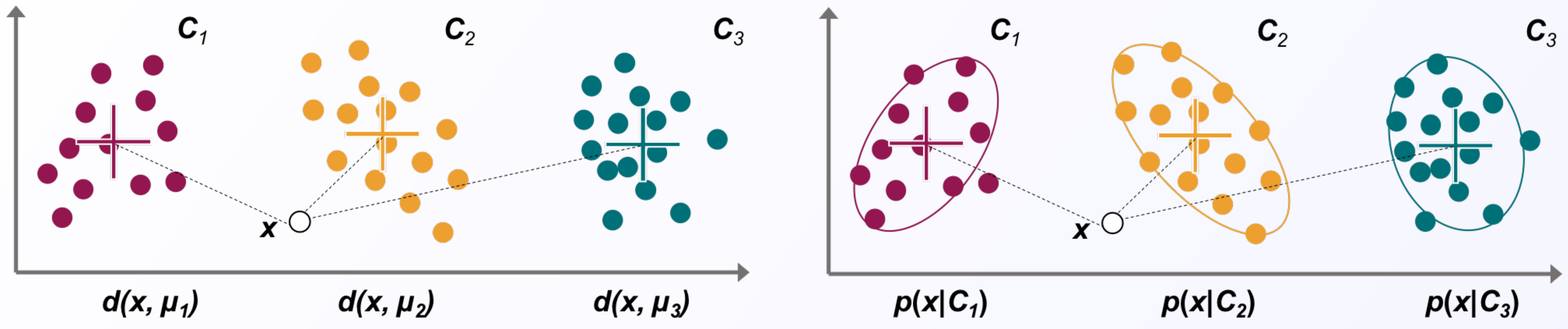


The above flow diagram illustrates a binary split codebook generation.

Next

- We have dealt with
  - Dimensionality
  - Better ways to describe data
- Key concepts
  - Models with '*parameters*'
  - Cost function optimised on training data.
  - Choice of distance function
- Still open:
  - Probabilistic models

Deterministic vs Probabilistic



- **Deterministic** models basically just assign new observations to one group or the other
- **Probabilistic** models (or stochastic models) will learn the probability distribution over the set of classes and use that to make predictions

Acoustic modelling

Remember: Speech recognition is based on the fundamental equation

$$\hat{W} = \arg\max_{W} P(A|W)P(W)$$

The acoustic model $P(A|W)$ estimates the match between the acoustics A and the word sequence W. The front-end converts the acoustic signal into a sequence of feature vectors (e.g. MFCCs):

$$A \equiv \mathbf{O} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T] = \mathbf{o}_1^T$$

where $\mathbf{o}_i$ are the feature vectors. The number of feature vectors T extracted from the audio signal varies due to the length of the utterance (e.g. one vector every 10 milliseconds). For many applications they consist of the 39 dimensional feature vectors outlined in the previous handout. Thus the probability computed by the acoustic model can now be written as

$$P(\mathbf{O}|W) = P\left(\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T|W\right)$$

This expression can be interpreted a **distance between the sound and the word**. However such a distance is still very complicated expression and needs simplification.

Independence !

The chain rule allows to split the probability term into a product of probabilities, one probability per frame. However, for each frame we have the previous frames as conditional variables (i.e. the part to the right of $'|'$). For many applications this would lead to intractable problems.

Thus for many applications frame independence is **assumed,**

$$P(\mathbf{o}_t|\mathbf{o}_1^t, M) \equiv P(\mathbf{o}_t|M)$$

which allows to simplify the computation even further:

$$P(\mathbf{O}|M) = \prod_{i=1}^{T} P(\mathbf{o}_t|M)$$

This means that **for each frame a score** (the probability) **is computed**. All scores are multiplied to yield the overall score.

What happens when we take the logarithm of the above ?

‘Maturity’ detection

Many other speech technology tasks work in a similar to speech recognition. This implies that a model is trained to map the acoustic signal to the target value of the classification task. For example one can automatically detect the gender of a speaker, male or female. In this case the acoustic model model is required to compute two probabilities:

$$P(\mathbf{O}|'child')$$
$$P(\mathbf{O}|'adult')$$

The detection (=recognition) operation is then performed by comparing these probabilities and selecting the gender with the highest probability. Using the **chain rule** we can write ($M =$ “*child*”):

$$P(\mathbf{O}|M) = P(\mathbf{o}_1|M)P(\mathbf{o}_2|\mathbf{o}_1, M)P(\mathbf{o}_3|\mathbf{o}_2, \mathbf{o}_1, M) \cdot \ldots = \prod_{i=1}^{T} P(\mathbf{o}_t|\mathbf{o}_1^t, M)$$

Naturally the same holds true for the “*adult*” part.

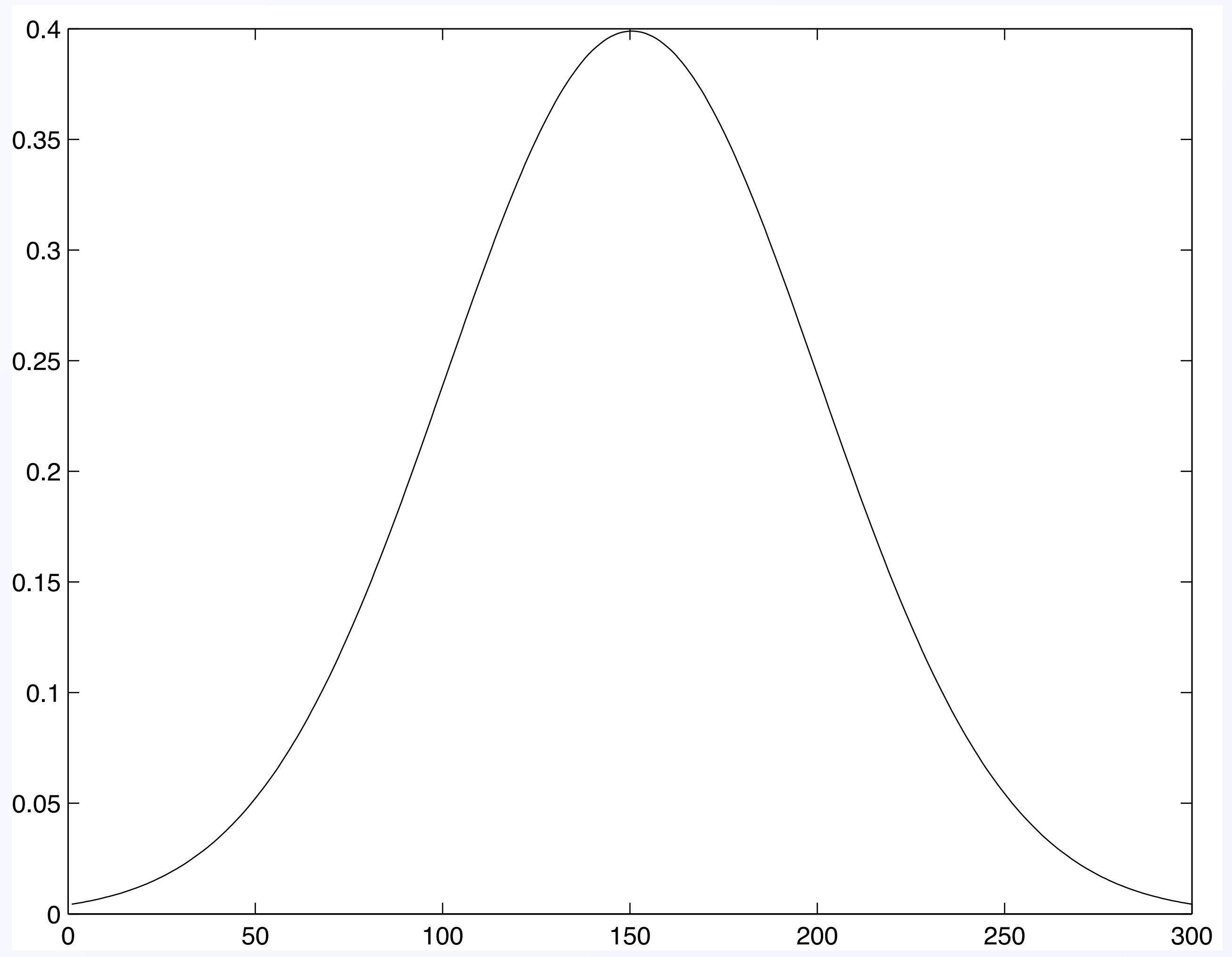
Gaussian models for (development/gender) classifcation

- A Gaussian model of the data (1D)
  - Parameters (mean, variance)

$$p_c(o) \equiv p(o|\lambda_c) \qquad p_a(o) \equiv p(o|\lambda_a)$$

- Similar to before, one model for each class is required
  - **Different to the deterministic case**
    The model includes the distance function

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



Frame by frame classification

- Decision theory requires to maximise posteriors
  - However, if we do not have prior information this amounts to a comparison of likelihood values.

$$P(c|o_t) \lesseqgtr P(a|o_t) \qquad \Leftrightarrow \qquad p_c(o_t) \lesseqgtr p_a(o_t)$$

- It is much more efficient to compare the log likelihood values

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

Why can we take the log ?

$$\log p(x) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(x-\mu)^2$$

Frame by frame classification

- The difference in log-likelihoods allows to perform the decision

$$\log p_c(o_t) - \log p_a(o_t) \propto \mathtt{const} + C_a(x-\mu_a)^2 - C_c(x-\mu_c)^2$$

This reminds of something ?

- For sequences the same applies:

$$\log\left(\prod_{t=1}^{T} p_c(o_t)\right) - \log\left(\prod_{t=1}^{T} p_a(o_t)\right)$$

- Of course this is also valid for high dimensions
  - Multi-variate Gaussian models

Gaussian models - Maximum likelihood

How can we obtain the ’example’ parameters?