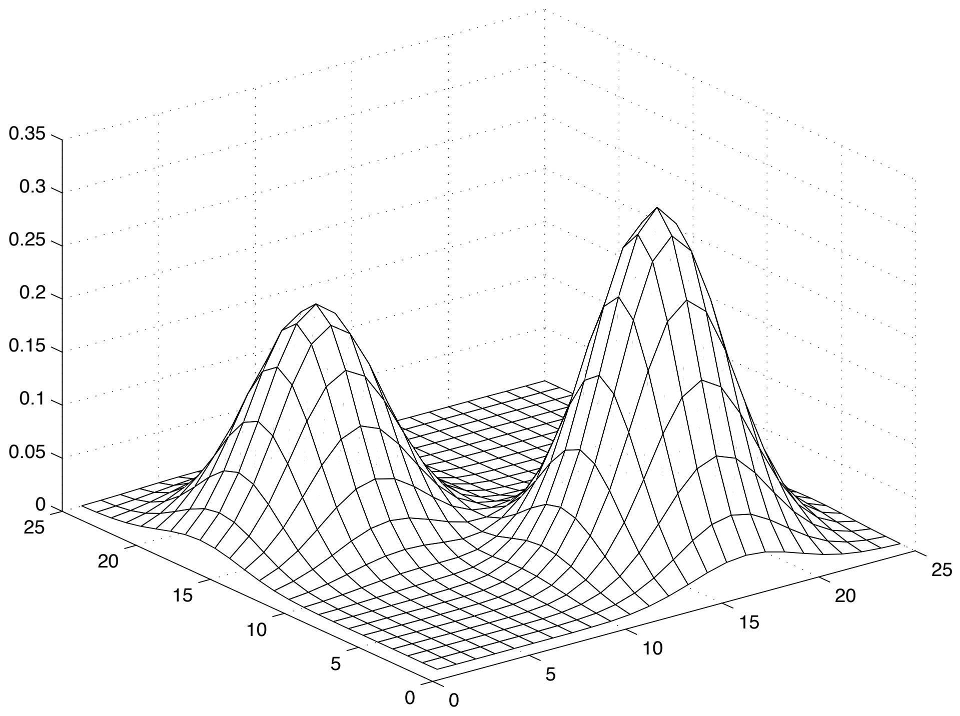
University of Cambridge
Engineering Part IIB

Module 4F10: Statistical Pattern Processing

Handout 3: Gaussian Mixture Models



Mark Gales
mjfg@eng.cam.ac.uk
Michaelmas 2013

Popular Generative Classifier

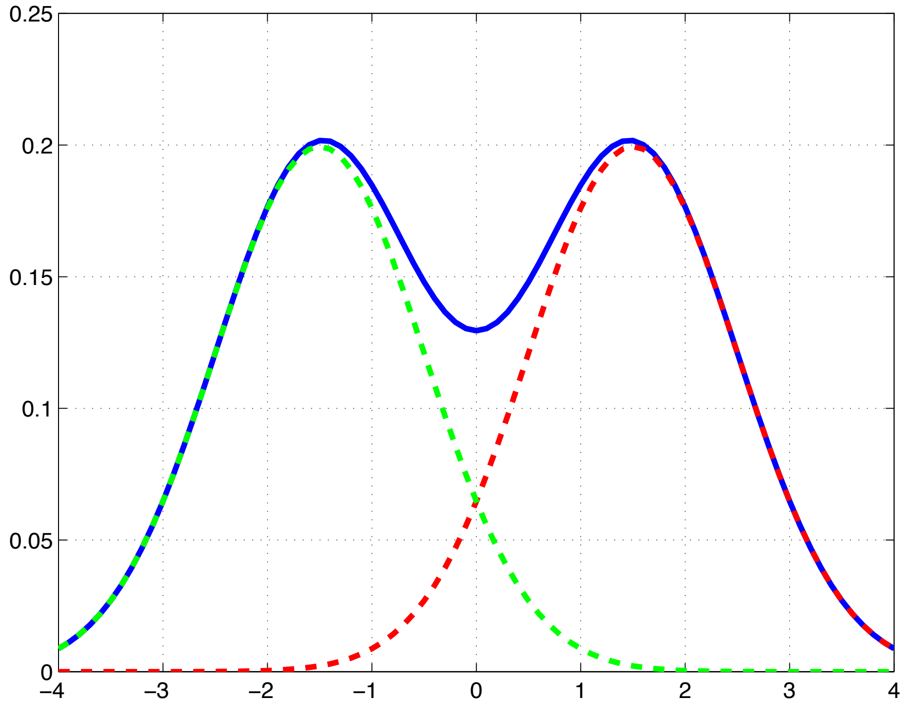
- A Study of Musical Instrument Classification Using Gaussian Mixture Models and Support Vector Machines
- Gaussian Mixture Model Classifiers for Machine Monitoring
- Gaussian Mixture Model Classification of Odontocetes in the Southern California Bight and the Gulf of California
- Texture Classification and Segmentation using Wavelet Packet Frame and Gaussian Mixture Model
- Classification of SNP Genotypes by a Gaussian Mixture Model in a Competitive Enzymatic Assays
- A Gaussian Mixture Model Classification Scheme for Myoelectric Control of Powered Upper Limb Prostheses
- Approaches to Language Identification using Gaussian Mixture Models and Shifted Delta Cepstral Features
- Confidence-Based Policy Learning from Demonstration Using Gaussian Mixture Models
- Classification of Facial Features using Gaussian Mixture Models

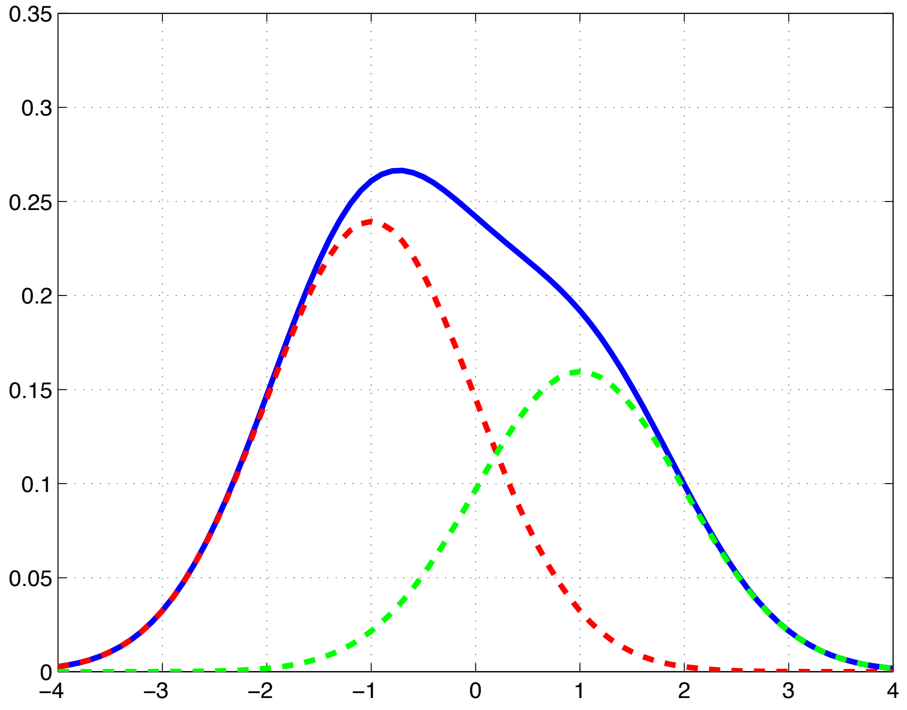
The final lecture in the course will discuss Speaker Verification using Gaussian Mixture Models (and SVMs for improved performance)

Probability of Error

$$\begin{aligned} P(\text{error}) &= P(\boldsymbol{x} \in \Omega_2, \omega_1) + P(\boldsymbol{x} \in \Omega_1, \omega_2) \\ &= P(\boldsymbol{x} \in \Omega_2 | \omega_1) P(\omega_1) + P(\boldsymbol{x} \in \Omega_1 | \omega_2) P(\omega_2) \\ &= \int_{\Omega_2} p(\boldsymbol{x}|\omega_1) P(\omega_1) d\boldsymbol{x} + \int_{\Omega_1} p(\boldsymbol{x}|\omega_2) P(\omega_2) d\boldsymbol{x} \end{aligned}$$

Gaussian Mixture Models

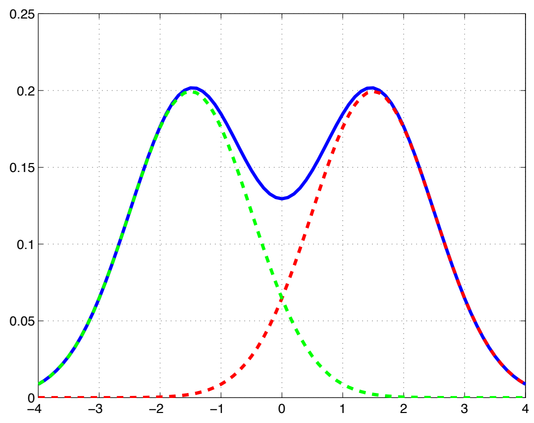


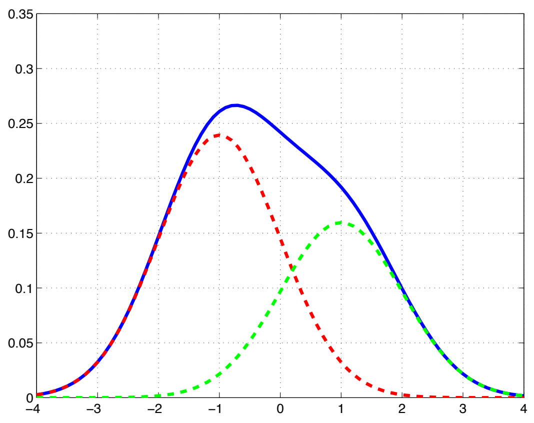


Introduction

The performance of a generative model is highly dependent on the accuracy of the class-conditional PDFs, $p(\boldsymbol{x}|\omega)$. For example, Gaussians do not well model

- multi-model and asymmetric data





Unfortunately data of this form is common. For instance in modelling a word in speech recognition

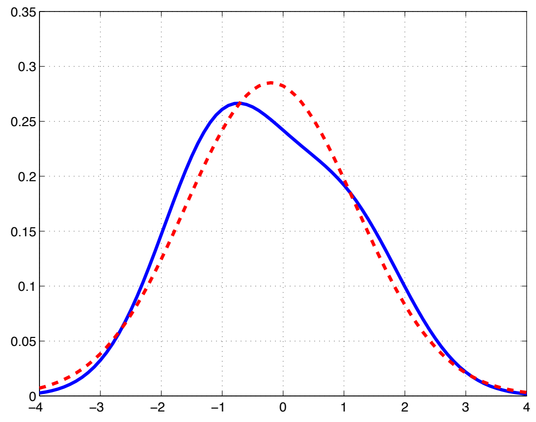
- male/female pronunciation differences
- accent variability (UK/American English)

These same type of effects often occur in other areas in which pattern recognition techniques are applied.

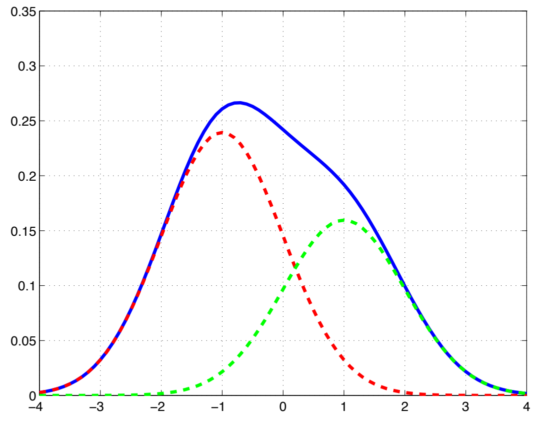
A general approach for modelling complicated distributions is mixture modelling. The form considered in this lecture is Gaussian mixture models (GMM). Under mild constraints these can be shown to be able to model any PDF.

Simple Example

Noise is generated by one of two sources. 60% of the time it is generated by a Gaussian distribution of mean -1 and variance 1. 40% of the time it is generate by a Gaussian distribution of mean 1 and variance 1. What is the overall distribution of the noise observed?



If a single Gaussian is used as a model, there is a poor fit.



A weighted combination of two Gaussian PDFs fit the data "perfectly". This is a mixture model. What we are interested in is the form and how to automatically train parameters of this mixture model.

Mixture Models

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{m=1}^{M} p(\boldsymbol{x}, \omega_m|\boldsymbol{\theta}_m) = \sum_{m=1}^{M} c_m p(\boldsymbol{x}|\omega_m, \boldsymbol{\theta}_m)$$

For a Gaussian mixture we have

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{m=1}^{M} c_m \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$$

Gaussian Mixture Models

The general form of a mixture model is

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{m=1}^{M} p(\boldsymbol{x}, \omega_m|\boldsymbol{\theta}_m) = \sum_{m=1}^{M} c_m p(\boldsymbol{x}|\omega_m, \boldsymbol{\theta}_m)$$

For a Gaussian mixture we have

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{m=1}^{M} c_m \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$$

where c_m is the component prior of each Gaussian component. For this to be a valid probability density function it is necessary that

$$\sum_{m=1}^{M} c_m = 1 \quad \text{and} \quad c_m \geq 0$$

The GMM parameters can be split into two parts

- component priors - $M-1$ parameters, $c_1, \ldots, c_M$
- component PDF - M times the parameters of a single component, $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M\}$

Increasing the number of components will yield more parameters and a more powerful class-conditional PDF, leading to a more powerful classifier

- need to be careful about correctly tuning the number of components

Mixture models can be used with other forms of PDF (and PMF), or combinations of different forms of PDF. This lecture will concentrate on GMMs.

Log-Likelihood Training

$$\mathcal{D} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}; \quad \mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log p(\boldsymbol{x}_i|\boldsymbol{\theta})$$

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \left(\sum_{m=1}^{M} c_m \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \right)$$

Log-Likelihood Function

We will estimate the parameters of a Gaussian mixture model using maximum likelihood (note mixtures of other distributions could also be considered).

To do maximum likelihood estimation, the log likelihood function for the data needs to be specified

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log p(\boldsymbol{x}_i|\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \left[\sum_{m=1}^{M} c_m p(\boldsymbol{x}_i|\omega_m, \boldsymbol{\theta}_m) \right]$$

where the dependence on the PDF for mixture component ω_m is explicit.

For ease of presentation, consider Gaussian distributions of the form $\mathbf{\Sigma}_m = \sigma_m^2 \mathbf{I}$, although the principles can be easily extended to more general covariance matrices.

Therefore

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \left[\sum_{m=1}^{M} \frac{c_m}{(2\pi\sigma_m^2)^{d/2}} \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{\mu}_m\|^2}{2\sigma_m^2} \right) \right]$$

Now, it is necessary to find the partial derivative of $\mathcal{L}(\boldsymbol{\theta})$ with respect to the parameters of the mixture distribution.

Log-Likelihood Derivative

Due to the form of the log likelihood we will (during the derivation below) use the substitution (from Bayes' noting that the c_m is a prior probability)

$$P(\omega_m|\boldsymbol{x}_i,\boldsymbol{\theta}) = \frac{c_m p(\boldsymbol{x}_i|\omega_m,\boldsymbol{\theta})}{p(\boldsymbol{x}_i|\boldsymbol{\theta})} = \frac{c_m p(\boldsymbol{x}_i|\omega_m,\boldsymbol{\theta})}{\sum_{j=1}^{M} c_j p(\boldsymbol{x}_i|\omega_j,\boldsymbol{\theta})}$$

where $P(\omega_m|\boldsymbol{x}_i,\boldsymbol{\theta})$ is the posterior probability of mixture component ω_m being associated with vector $\boldsymbol{x}_i$.

Considering a particular parameter θ_m that is associated with (only) the m^{th} mixture component.

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_m} = \sum_{i=1}^{n} \frac{1}{p(\boldsymbol{x}_i|\boldsymbol{\theta})} \frac{\partial \left[c_m p(\boldsymbol{x}_i|\omega_m,\boldsymbol{\theta}_m)\right]}{\partial \boldsymbol{\theta}_m}$$

c_m is not a function of $\boldsymbol{\theta}_m$. Consider just the mean of component ω_m

$$\begin{aligned}\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\mu}_m} &= \sum_{i=1}^{n} \frac{c_m \mathcal{N}(\boldsymbol{x}_i|\omega_m,\boldsymbol{\theta}_m)}{p(\boldsymbol{x}_i|\boldsymbol{\theta})} \frac{(\boldsymbol{x}_i - \boldsymbol{\mu}_m)}{\sigma_m^2} \\ &= \sum_{i=1}^{n} P(\omega_m|\boldsymbol{x}_i,\boldsymbol{\theta}) \frac{(\boldsymbol{x}_i - \boldsymbol{\mu}_m)}{\sigma_m^2}\end{aligned}$$

Unlike a Gaussian, no closed-form solution, as $P(\omega_m|\boldsymbol{x}_i,\boldsymbol{\theta})$ is a non-linear function of the mean $\boldsymbol{\mu}_m$.

Gradient descent (or related schemes) could be used, but anything better?

"Incorrect" Updates

Initially ignore the dependence of the posterior $P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta})$ on the component parameters $\boldsymbol{\theta}_m$, equating to zero yields to find the estimate of the mean, $\hat{\boldsymbol{\mu}}_m$,

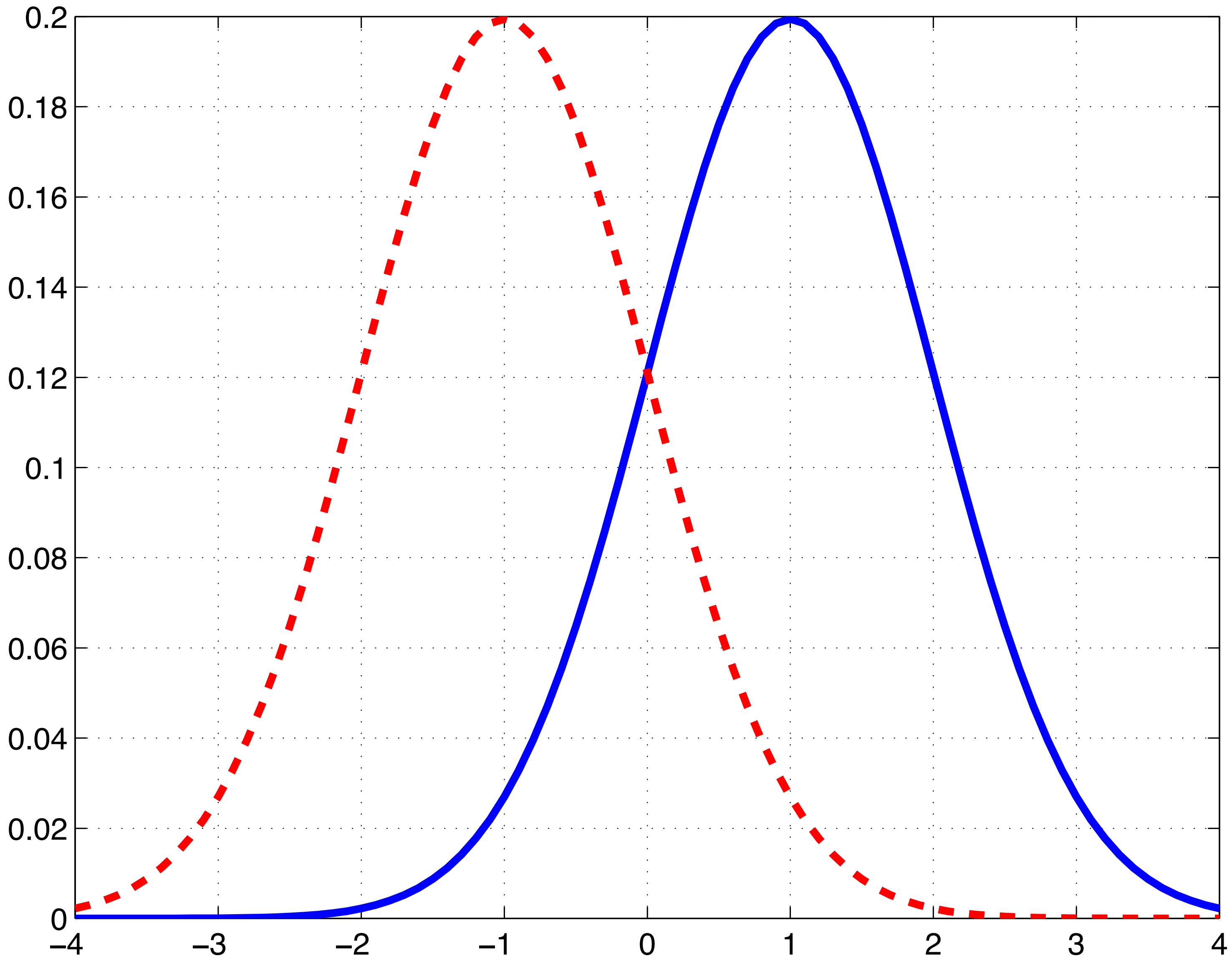
$$\begin{aligned}
0 &= \sum_{i=1}^{n} P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta})\frac{(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_m)}{\sigma_m^2} \\
\frac{1}{\sigma_m^2}\sum_{i=1}^{n} P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta})\hat{\boldsymbol{\mu}}_m &= \frac{1}{\sigma_m^2}\sum_{i=1}^{n} P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta})\boldsymbol{x}_i \\
\hat{\boldsymbol{\mu}}_m &= \frac{\sum_{i=1}^{n} P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta})\boldsymbol{x}_i}{\sum_{i=1}^{n} P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta})}
\end{aligned}$$

Repeating the process for the variance term σ_m^2 yields an update of

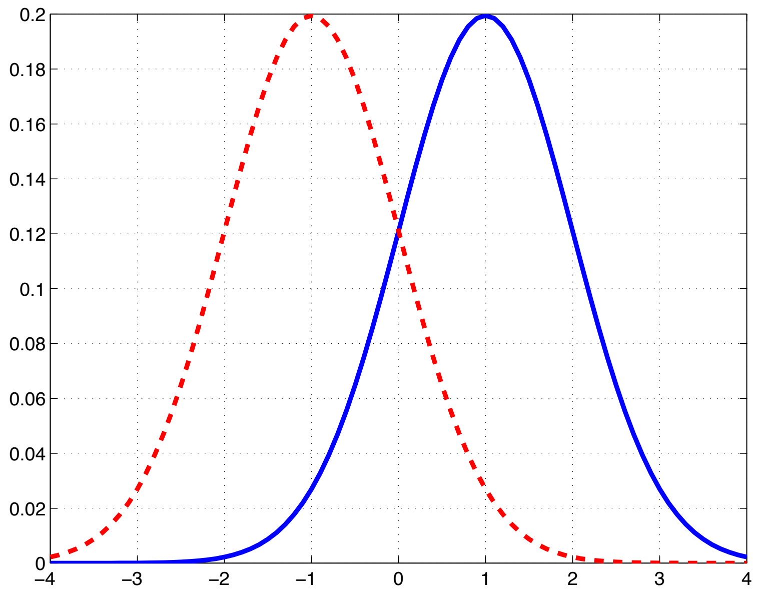
$$\hat{\sigma}_m^2 = \frac{1}{d}\frac{\sum_{i=1}^{n} P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta})\left|\left|\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_m\right|\right|^2}{\sum_{i=1}^{n} P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta})}$$

Unfortunately these estimates are only correct when the estimates used to get the posterior $P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta})$ is the same as the new estimates - must be at a local maximum!

In general it is not possible to guarantee that the log-likelihood function, $\mathcal{L}(\boldsymbol{\theta})$, will increase.


0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
–4
–3
–2
–1
0
1
2
3
4

Simple "EM"



The above diagram shows two Gaussian components, each having an equal prior ($c_1 = c_2 = P(\omega_1) = P(\omega_2) = 0.5$). We have single dimensional training data $x_1, \ldots, x_n$.

If the correct component for each observation was known we could use the standard estimation formulae for each of the components.

- BUT we don't know the correct component.
- BUT we can estimate the assignment using the Gaussians above.

This is a classification problem, so use Bayes' to assign it.

Simple "EM" (cont)

Let the assignment variable be z. For the above example we label all the data using

$$z_i = \begin{cases} \omega_1 & \text{if } x_i > 0 \\ \omega_2 & \text{if } x_i < 0 \end{cases}$$

The the estimate of the mean is then

$$\hat{\mu}_1 = \frac{1}{n_1} \sum_{z_i = \omega_1} x_i$$

n_1 is the number of samples assigned to ω_1. The variance is

$$\hat{\sigma}_1^2 = \frac{1}{n_1} \sum_{z_i = \omega_1} (x_i - \hat{\mu}_1)^2$$

The prior can be estimated simply by using the relative frequencies of class ω_1 and ω_2

$$\hat{c}_1 = \frac{n_1}{n_1 + n_2}$$

Similarly for class ω_2.

We now have new estimates of the component parameters. We can therefore get new estimates of the assignment z. We can carry on doing this loop until nothing changes.

This is a simple iterative process that is guaranteed not to decrease the likelihood.

Lagrange Optimisation

To estimate the component priors from the maximum of the log likelihood function, we note the constraint that the component priors must sum to one and be positive. This can be done using the method of Lagrange multipliers[1].

- Assume a extremum (maximum/minimum) of a scalar valued function $f(\boldsymbol{x})$ is required subject to a constraint.
- If the constraint can be expressed as $g(\boldsymbol{x}) = c$ then we can transform the constrained optimisation into an unconstrained one by finding the extremum of the Lagrangian function
$$L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda \left[g(\boldsymbol{x}) - c \right]$$
- λ is called the Lagrange multiplier
- Find for extremum
$$\frac{\partial L(\boldsymbol{x}, \lambda)}{\partial \boldsymbol{x}} = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} + \lambda \frac{\partial \left[g(\boldsymbol{x}) - c \right]}{\partial \boldsymbol{x}} = 0$$
  Solve to give the required value of $\boldsymbol{x}$ and λ and hence extremum of $f(\boldsymbol{x})$ subject to the constraint $g(\boldsymbol{x}) = c$.
- Note also that
$$\frac{\partial L(\boldsymbol{x}, \lambda)}{\partial \lambda} = 0$$
  when the constraint is satisfied.
- We will use this method in several places in this course.

[1]see Bishop (1995) p.64 for an alternative method

“Incorrect” Prior Updates

The constraint to add is that $\sum_{m=1}^{M} c_m = 1$. In this case add $\lambda(\sum_{m=1}^{M} c_m - 1)$, so the Lagrangian to optimise is

$$L(\boldsymbol{\theta}, \lambda) = \mathcal{L}(\boldsymbol{\theta}) + \lambda(\sum_{m=1}^{M} c_m - 1)$$

Differentiating this

$$\begin{aligned}\frac{\partial\left(L(\boldsymbol{\theta},\lambda)\right)}{\partial c_m} &= \sum_{i=1}^{n} \frac{P(\omega_m|\boldsymbol{x}_i,\boldsymbol{\theta})}{c_m} - \lambda \\ \frac{\partial\left(L(\boldsymbol{\theta},\lambda)\right)}{\partial \lambda} &= \sum_{m=1}^{M} c_m - 1\end{aligned}$$

Ignoring the dependence of $P(\omega_m|\boldsymbol{x}_i,\boldsymbol{\theta})$ on c_m, then the update (satisfy the first differential)

$$\hat{c}_m = \frac{1}{\lambda}\sum_{i=1}^{n} P(\omega_m|\boldsymbol{x}_i,\boldsymbol{\theta})$$

Satisfying the second differential ($\sum_{m=1}^{M} c_m = 1$) requires

$$\sum_{m=1}^{M}\frac{1}{\lambda}\sum_{i=1}^{n} P(\omega_m|\boldsymbol{x}_i,\boldsymbol{\theta}) = 1, \quad \lambda = \sum_{i=1}^{n}\sum_{m=1}^{M} P(\omega_m|\boldsymbol{x}_i,\boldsymbol{\theta}) = n$$

so

$$\hat{c}_m = \frac{1}{n}\sum_{k=1}^{n} P(\omega_m|\boldsymbol{x}_i,\boldsymbol{\theta})$$

Parameter Estimation

The previous process has performed a "hard" assignment of observation to component. This can be generalised to a probabilistic assignment. This form of update is then referred to as an example of the Expectation-Maximisation algorithm. The theory behind this will be discussed in the next lecture. EM is an iterative process, so let the parameters at iteration k be denoted by $\boldsymbol{\theta}^{(k)}$

$$\begin{aligned}
\boldsymbol{\mu}_m^{(k+1)} &= \frac{\sum_{i=1}^n P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta}^{(k)})\boldsymbol{x}_i}{\sum_{i=1}^n P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta}^{(k)})} \\
\sigma_m^{(k+1)2} &= \frac{1}{d}\frac{\sum_{i=1}^n P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta}^{(k)})\left\|\boldsymbol{x}_i - \boldsymbol{\mu}_m^{(k+1)}\right\|^2}{\sum_{i=1}^n P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta}^{(k)})} \\
c_m^{(k+1)} &= \frac{1}{n}\sum_{i=1}^n P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta}^{(k)})
\end{aligned}$$

The more general update for the covariance matrix is

$$\boldsymbol{\Sigma}_m^{(k+1)} = \frac{\sum_{i=1}^n P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta}^{(k)})(\boldsymbol{x}_i - \boldsymbol{\mu}_m^{(k+1)})(\boldsymbol{x}_i - \boldsymbol{\mu}_m^{(k+1)})'}{\sum_{k=1}^n P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta}^{(k)})}$$

The application of these equations is guaranteed to provide an increase in the likelihood function unless the likelihood function is at a local maximum (proof of E-M next lecture ...). Thus

$$\mathcal{L}(\boldsymbol{\theta}^{(k+1)}) \geq \mathcal{L}(\boldsymbol{\theta}^{(k)})$$

EM Training Procedure

Each iteration of the E-M algorithm for Gaussian Mixtures operates in two stages:

1. Find the posterior probability of mixture component occupation using the current parameter values.
2. Update the parameters of the Gaussian mixture as though the posterior probabilities were the true values.

Therefore the overall procedure is

1. Initialise the parameters of the mixture, $\boldsymbol{\theta}^{(0)}$, - for example set all component priors to be equal, all variances to be equal, and use different values for the mean vectors and set $k = 0$
2. Compute $P(\omega_m|\boldsymbol{x}_i, \boldsymbol{\theta}^{(k)})$ for every data point and accumulate the statistics for the numerators and denominators of the re-estimation formulae. Also compute the log likelihood of the data set
3. Update the parameters as necessary to give $\boldsymbol{\theta}^{(k+1)}$, set $k = k + 1$
4. If the log likelihood increase is less than a threshold stop, else goto 2.

Note that only a local maximum of the likelihood function is found by this procedure so the initialisation of the scheme is important, and can have problems (e.g. a variance can tend to zero and likelihood become infinite!)

Simple Worked Example

Consider some data from 2 classes:

Class 1 has points $\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}0.6\\0.6\end{bmatrix}, \begin{bmatrix}0.7\\0.4\end{bmatrix}$

Class 2 has points $\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0.25\\1\end{bmatrix}, \begin{bmatrix}0.3\\0.4\end{bmatrix}$

The aim is to build a mixture model on the composite data. The variances of both components are fixed at the identity matrix. Initial values of the means are

$$\boldsymbol{\mu}_1 = \begin{bmatrix}0.25\\0.25\end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix}0.75\\0.75\end{bmatrix}$$

First the posteriors for the two model sets are required.
Class 1 has posteriors :

Comp1	0.5	0.3775	0.4750	0.4875
Comp2	0.5	0.6225	0.5250	0.5125

Class 2 has posteriors:

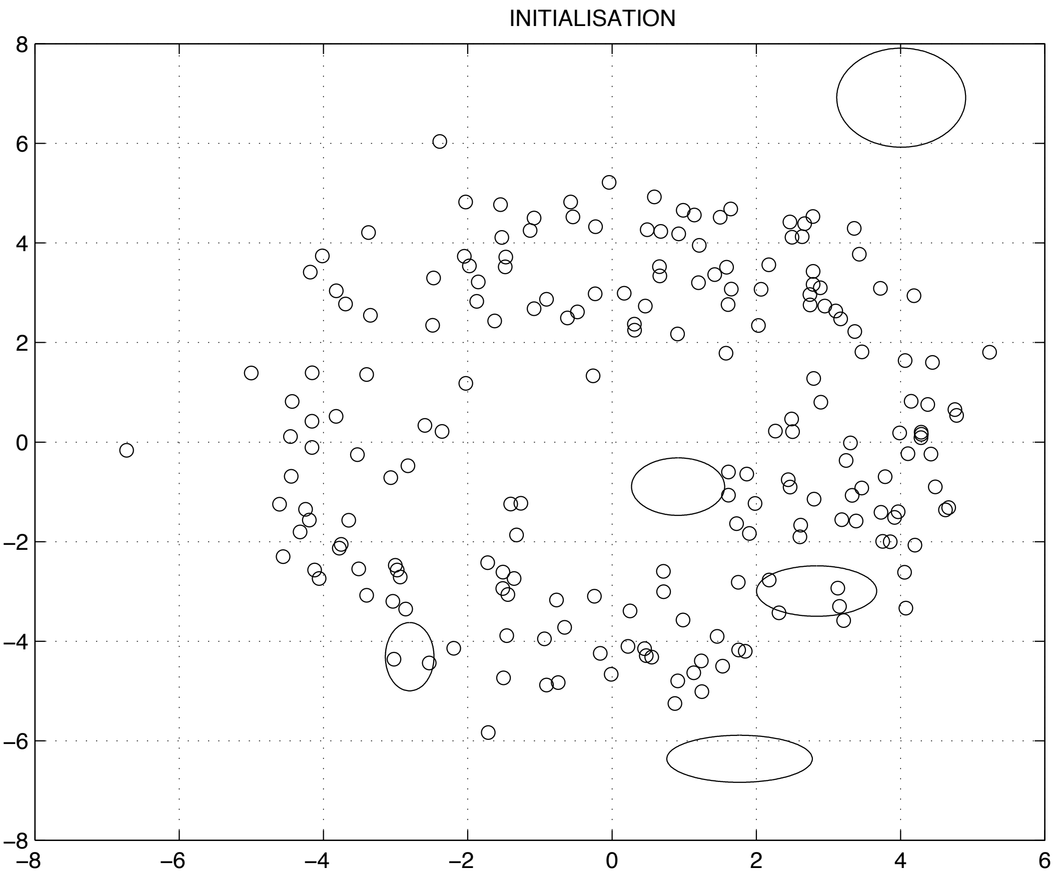
Comp1	0.6225	0.5	0.4688	0.5374
Comp2	0.3775	0.5	0.5312	0.4626

This gives updated means of:

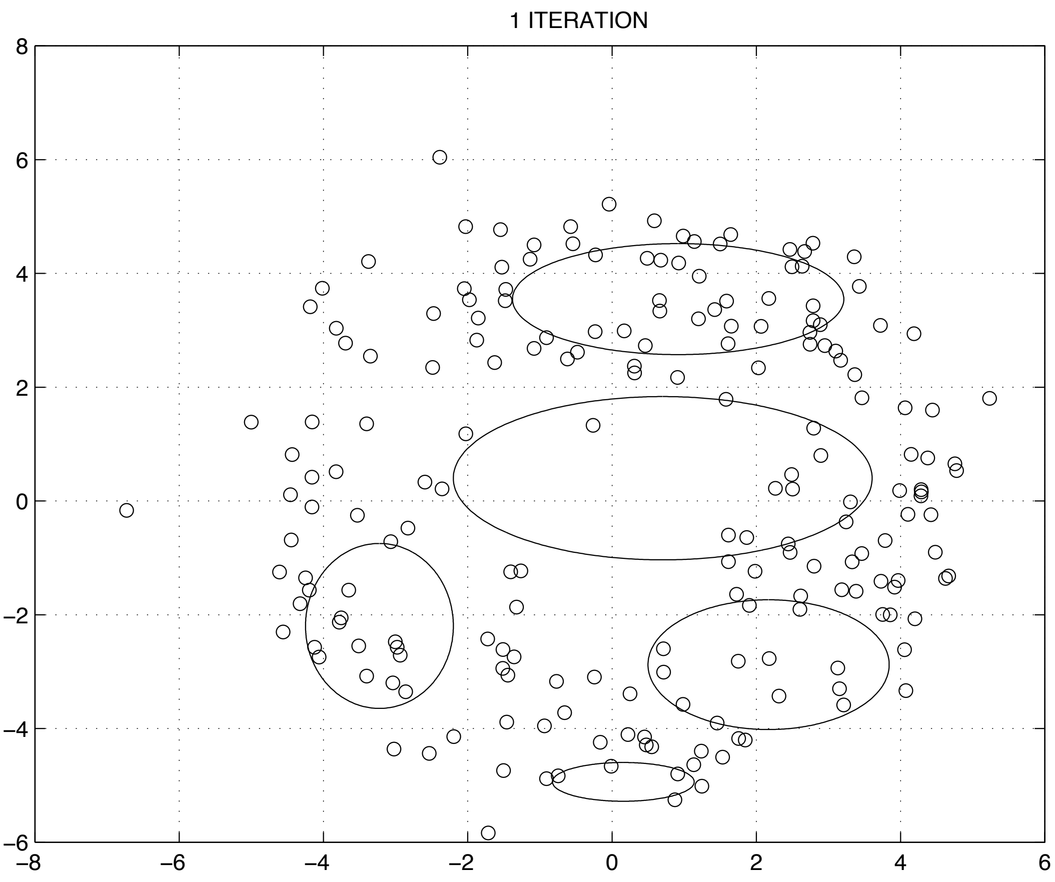
$$\boldsymbol{\mu}_1 = \begin{bmatrix}0.4491\\0.5143\end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix}0.5129\\0.5851\end{bmatrix}$$

This model has an increased likelihood of generating the data. Further iterations will increase the likelihood further.

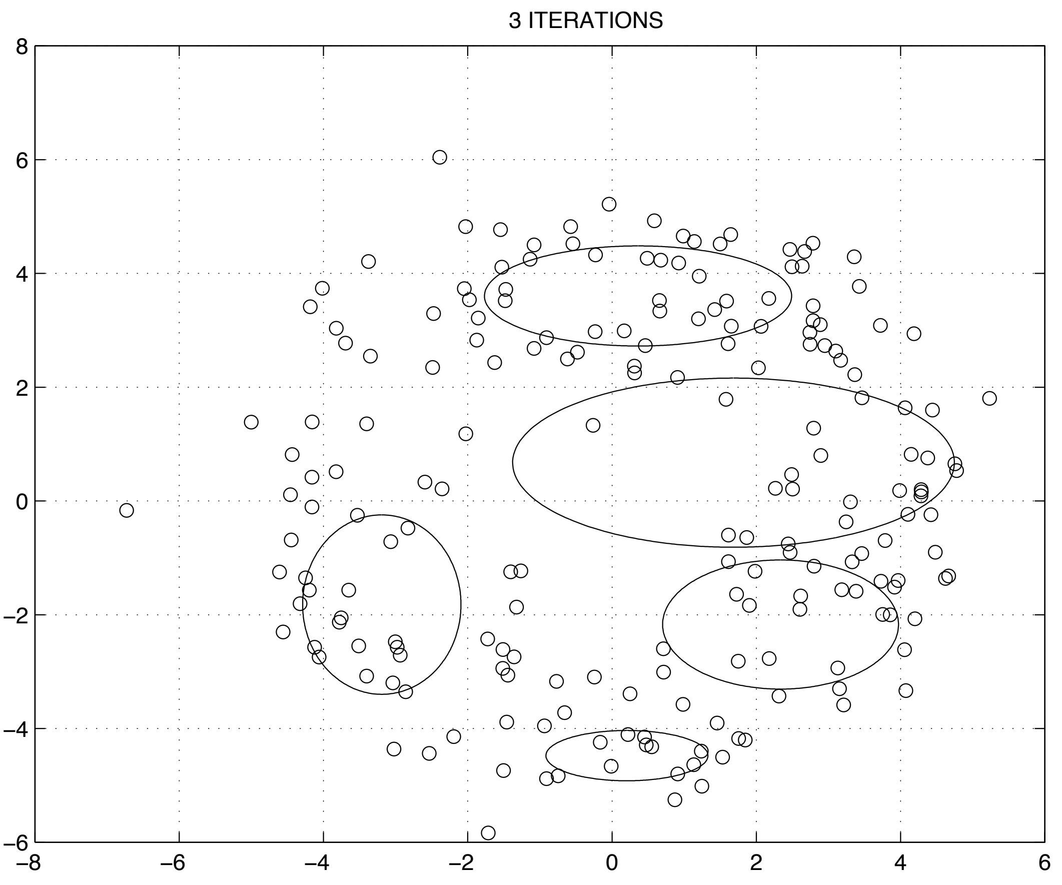
Further Example initialisation



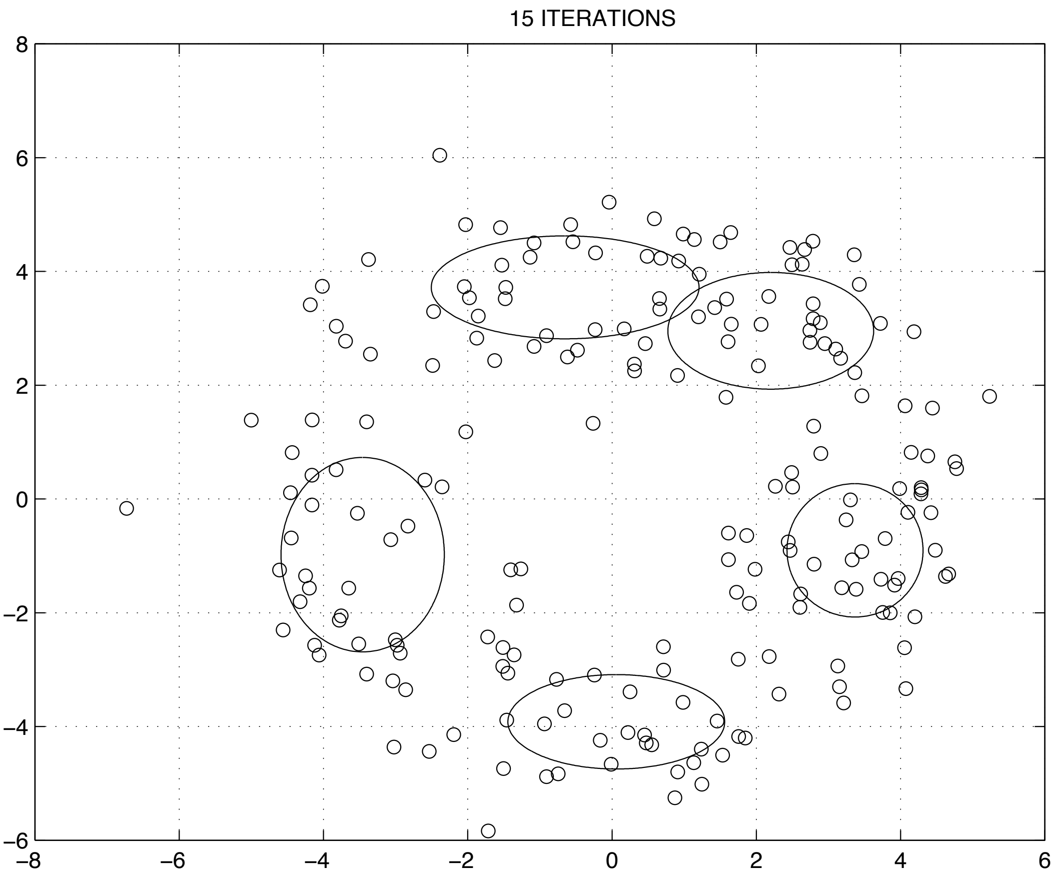
Further Example - 1 iteration



Further Example - 3 iterations

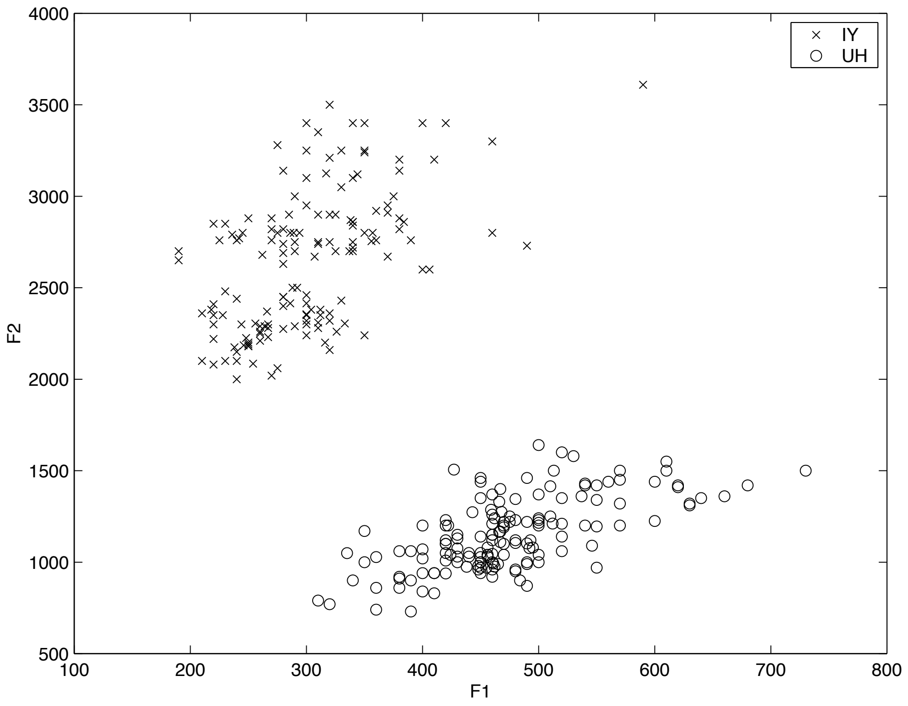


Further Example - 15 iterations



Extracting Structure

Rather than using GMMs to model arbitrary PDFs, they may be used to extract structure from unlabelled data. Consider some simple speech data from which the formants (F1 and F2) have been extracted and the hard GMM assignment.



A two component GMM could be used to model the data. The hope is that the components will model different classes of the underlying data. Though it is not possible to label what the class component labels are it can be used to decide if data is similar to one class or another - clustering

The problems are:

- how many components to use;
- the final solution depends on the initialisation!