

Gaussian Mixture Models

Peter Bell

Automatic Speech Recognition— ASR Lecture 6
1 February 2024

GMMs

- Univariate and multivariate Gaussians
- Gaussian mixture models
- GMM estimation with the EM algorithm
- Using GMMs with HMMs

Consider a real valued random variable X

- Cumulative distribution function (cdf) $F(x)$ for X :

$$F(x) = P(X \leq x)$$

- To obtain the probability of falling in an interval we can do the following:

$$\begin{aligned} P(a < X \leq b) &= P(X \leq b) - P(X \leq a) \\ &= F(b) - F(a) \end{aligned}$$

- The rate of change of the cdf gives us the *probability density function* (pdf), $p(x)$:

$$p(x) = \frac{d}{dx}F(x) = F'(x)$$

$$F(x) = \int_{-\infty}^x p(x)dx$$

- $p(x)$ is **not** the probability that X has value x . But the pdf is proportional to the probability that X lies in a small interval centred on x .
- Notation: p for pdf, P for probability

The Gaussian distribution (univariate)

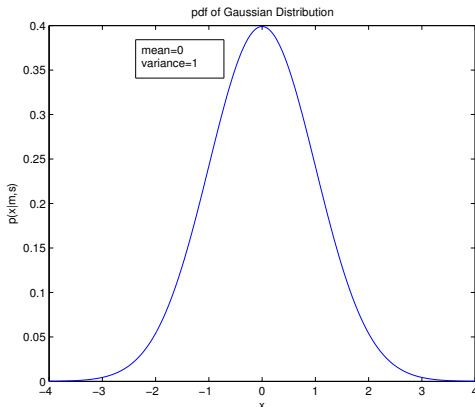
- The **Gaussian** (or **Normal**) distribution is the most common (and easily analysed) continuous distribution
- It is also a reasonable model in many situations (the famous “bell curve”)
- If a (scalar) variable has a Gaussian distribution, then it has a probability density function with this form:

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

- The Gaussian is described by two parameters:
 - the mean μ (location)
 - the variance σ^2 (dispersion)

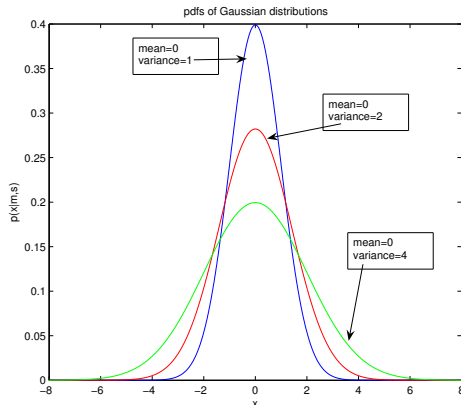
Plot of Gaussian distribution

- Gaussians have the same shape, with the location controlled by the mean, and the spread controlled by the variance
- One-dimensional Gaussian with zero mean and unit variance ($\mu = 0, \sigma^2 = 1$):



Properties of the Gaussian distribution

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



Parameter estimation

- Estimate mean and variance parameters of a Gaussian from data x_1, x_2, \dots, x_N
- Use the following as the estimates:

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n \quad (\text{mean})$$

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \hat{\mu})^2 \quad (\text{variance})$$

Example: ML estimation of the mean

Consider the log likelihood of a set of N training data points $\{x_1, \dots, x_N\}$ being generated by a Gaussian with mean μ and variance σ^2 :

$$\begin{aligned} L = \ln p(\{x_1, \dots, x_N\} | \mu, \sigma^2) &= -\frac{1}{2} \sum_{n=1}^N \left(\frac{(x_n - \mu)^2}{\sigma^2} - \ln \sigma^2 - \ln(2\pi) \right) \\ &= -\frac{1}{2\sigma^2} \sum_{n=1}^N (x_n - \mu)^2 - \frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi) \end{aligned}$$

By maximising the the log likelihood function with respect to μ we can show that the maximum likelihood estimate for the mean is indeed the sample mean:

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n.$$

The multivariate Gaussian distribution

- The D -dimensional vector $\mathbf{x} = (x_1, \dots, x_D)^T$ follows a multivariate Gaussian (or normal) distribution if it has a probability density function of the following form:

$$p(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

The pdf is parameterised by the mean vector $\boldsymbol{\mu} = (\mu_1, \dots, \mu_D)^T$ and the covariance matrix $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{11} & \dots & \sigma_{1D} \\ \vdots & \ddots & \vdots \\ \sigma_{D1} & \dots & \sigma_{DD} \end{pmatrix}$.

- The 1-dimensional Gaussian is a special case of this pdf
- The argument to the exponential $0.5(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$ is referred to as a *quadratic form*.

Covariance matrix

- The mean vector μ is the expectation of \mathbf{x} :

$$\mu = E[\mathbf{x}]$$

- The covariance matrix Σ is the expectation of the deviation of \mathbf{x} from the mean:

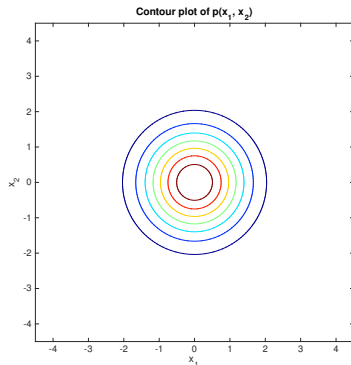
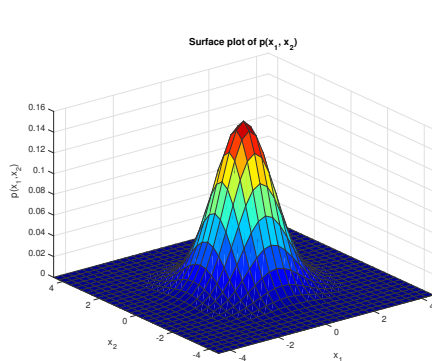
$$\Sigma = E[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T]$$

- Σ is a $D \times D$ symmetric matrix:

$$\sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)] = E[(x_j - \mu_j)(x_i - \mu_i)] = \sigma_{ji}$$

- The sign of the covariance helps to determine the relationship between two components:
 - If x_j is large when x_i is large, then $(x_i - \mu_i)(x_j - \mu_j)$ will tend to be positive;
 - If x_j is small when x_i is large, then $(x_i - \mu_i)(x_j - \mu_j)$ will tend to be negative.

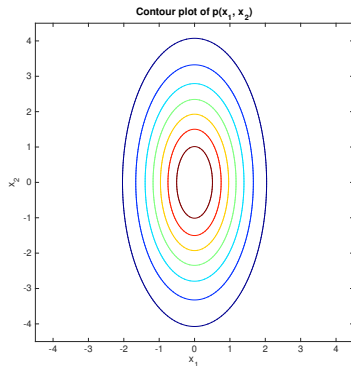
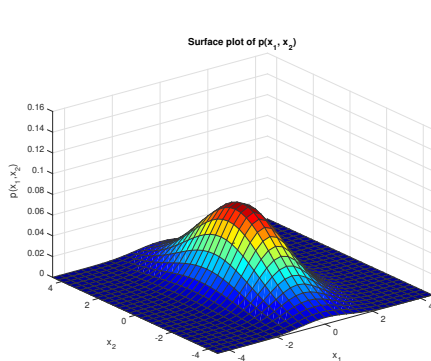
Spherical Gaussian



$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \rho_{12} = 0$$

NB: Correlation coefficient $\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} \quad (-1 \leq \rho_{ij} \leq 1)$

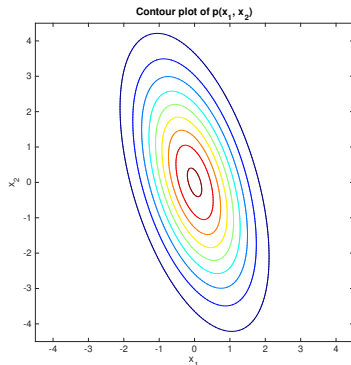
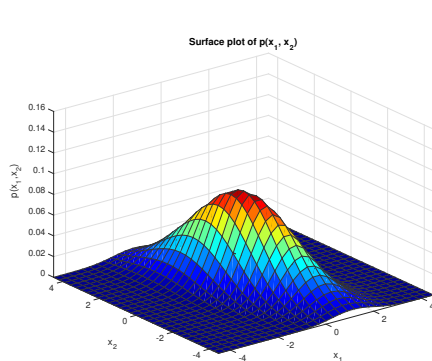
Diagonal Covariance Gaussian



$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix} \quad \rho_{12} = 0$$

NB: Correlation coefficient $\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} \quad (-1 \leq \rho_{ij} \leq 1)$

Full covariance Gaussian



$$\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad \Sigma = \begin{pmatrix} 1 & -1 \\ -1 & 4 \end{pmatrix} \quad \rho_{12} = -0.5$$

NB: Correlation coefficient $\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} \quad (-1 \leq \rho_{ij} \leq 1)$

Parameter estimation of a multivariate Gaussian distribution

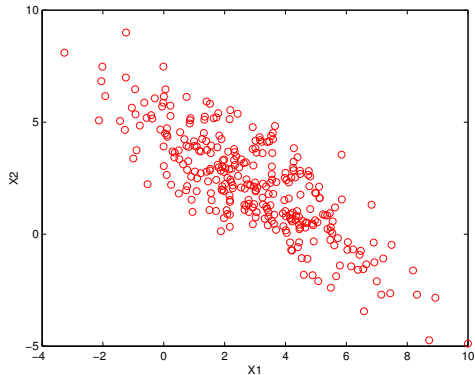
- It is possible to show that the mean vector $\hat{\mu}$ and covariance matrix $\hat{\Sigma}$ that maximise the likelihood of the training data are given by:

$$\hat{\mu} = \frac{1}{N} \sum_{t=1}^N \mathbf{x}_t$$
$$\hat{\Sigma} = \frac{1}{N} \sum_{t=1}^N (\mathbf{x}_t - \hat{\mu})(\mathbf{x}_t - \hat{\mu})^T$$

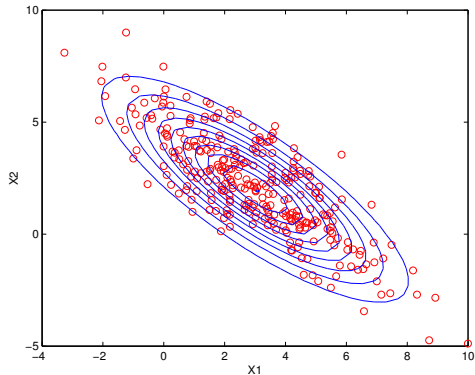
where $\mathbf{x}_n = (x_{n,1}, \dots, x_{n,D})^T$.

NB: T denotes either the number of samples or vector transpose depending on context.

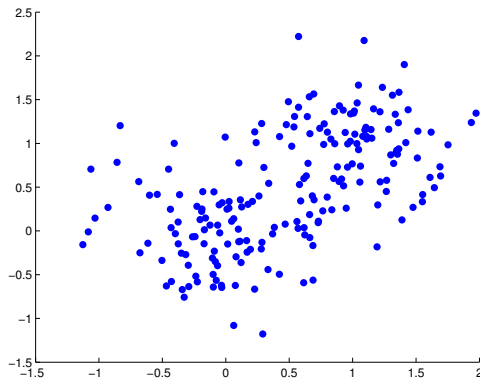
Example data



Maximum likelihood fit to a Gaussian

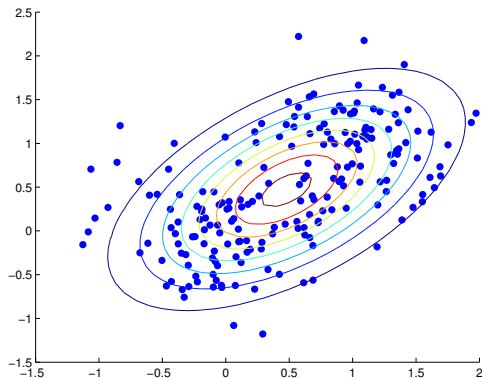


Data in clusters (example 1)



$$\mu_1 = (0, 0)^T \quad \mu_2 = (1, 1)^T \quad \Sigma_1 = \Sigma_2 = 0.2I$$

Example 1 fit by a Gaussian



$$\mu_1 = (0, 0)^T \quad \mu_2 = (1, 1)^T \quad \Sigma_1 = \Sigma_2 = 0.2I$$

Mixture model

- A more powerful form of density estimation is to introduce multiple *components* to the model, each with its own probability density. This is called a *mixture model* or a *mixture density*
- Can view this as a generative model
 - ① Choose a random mixture component C based on a prior probability $P(C = m)$
 - ② Generate a data point x from the chosen component using a density function $p(x|C = m)$

****The component C is not observed****

- We can calculate the probability density of x as

$$p(x) = \sum_{m=1}^M P(C = m)p(x|C = m)$$

- We use shorthand notation $P(m)$, $p(x|m)$

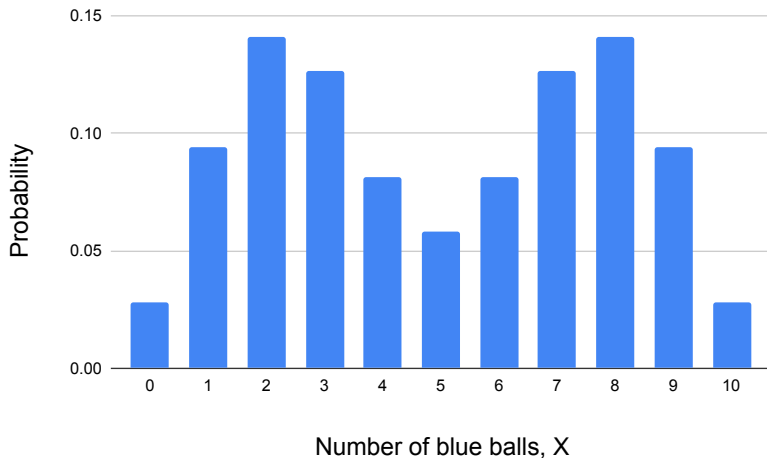
Simple example

- Suppose we have two identical bags, each containing a different proportion of blue balls. In each trial, we randomly chose a bag with probability 0.5 and pull out k balls (with replacement).
- What is the distribution of X , the number of blue balls sampled?

$$P(X = i) = \frac{1}{2} \text{Bin}(k, \alpha_1) + \frac{1}{2} \text{Bin}(k, \alpha_2)$$

where α_1, α_2 are the proportions of blue balls in the respective bags

Simple example



Example for $\alpha_1 = \frac{1}{4}$, $\alpha_2 = \frac{3}{4}$, $k = 10$

- The most important mixture model is the *Gaussian Mixture Model* (GMM), where the component densities are Gaussians

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

Estimating the parameters of a mixture model

- Define the indicator variable $z_{mn} = 1$ if component m generated data point x_n (and 0 otherwise)
- If z_{mn} wasn't hidden then we could count the number of observed data points generated by m :

$$N_m = \sum_{n=1}^N z_{mn}$$

- And use the observations assigned to each component to estimate the parameters using maximum likelihood estimation

In our simple example

- Suppose we repeat the experiment (trial) n times
- z_{mn} indicates if bag m was chosen on the n th trial
- If x_n is the number of blue balls drawn on the n th trial

$$\hat{\alpha}_m = \frac{\sum_{n=1}^N z_{mn} x_n}{k \times N_m}$$

- If the bags were not chosen with identical probability, we could estimate this probability with

$$\hat{P}(m) = \frac{1}{N} \sum_n z_{mn} = \frac{N_m}{N}$$

GMM Parameter estimation when we know which component generated the data

Estimate the mean, covariance and mixing parameters as:

$$\begin{aligned}\hat{\boldsymbol{\mu}}_m &= \frac{\sum_n z_{mn} \mathbf{x}_n}{N_m} \\ \hat{\boldsymbol{\Sigma}}_m &= \frac{\sum_n z_{mn} (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_m)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_m)^T}{N_m} \\ \hat{P}(m) &= \frac{1}{N} \sum_n z_{mn} = \frac{N_m}{N}\end{aligned}$$

Parameter estimation when we don't know which component generated the data

- *Problem:* we don't know z_{mn} - which mixture component a data point comes from...
- Instead we use the EM algorithm: estimate the posterior probability $P(m|x)$, which gives the probability that component m was responsible for generating data point x , using an initial set of parameters, λ_0
- At each iteration, we maximise

$$\sum_m P(m|x; \lambda_0) \log P(x, m; \lambda)$$

$$P(m|x; \lambda_0) = \frac{p(x|m) P(m)}{p(x)} = \frac{p(x|m) P(m)}{\sum_{m'=1}^M p(x|m') P(m')}$$

(dropping the dependence on λ_0 for clarity)

- We can view the EM algorithm as estimating “*soft counts*” for the data points, based on the component occupation probabilities $P(m|x_n)$:

$$N_m^* = \sum_{n=1}^N P(m|x_n)$$

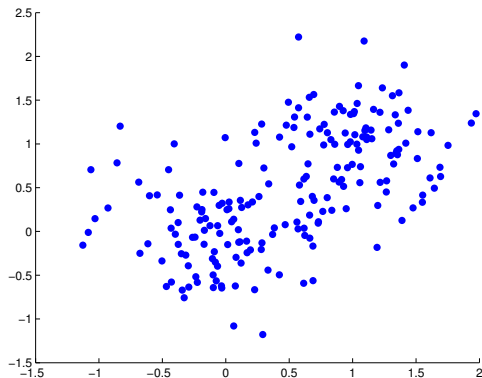
- We can imagine this as assigning data points to component m weighted by the component occupation probability $P(m|x_n)$
- In the bag example: imagine estimating which bag has been chosen at the n th trial, based on the number of blue balls drawn (and our earlier estimates of the parameters)
- It is possible to prove that the EM algorithm is guaranteed to increase the likelihood at each iteration

For the GMM

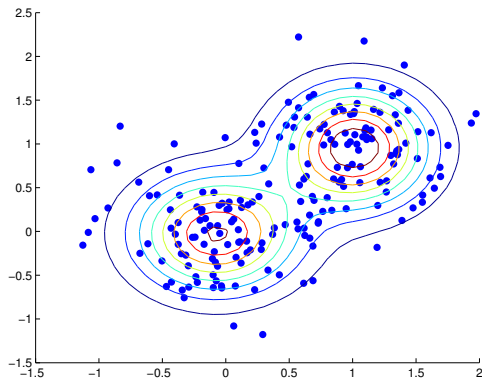
Estimate the mean, variance and prior probabilities as:

$$\begin{aligned}\hat{\mu}_m &= \frac{\sum_n P(m|\mathbf{x}_n)\mathbf{x}_n}{\sum_n P(m|\mathbf{x}_n)} = \frac{\sum_n P(m|\mathbf{x}_n)\mathbf{x}_n}{N_m^*} \\ \hat{\Sigma}_m &= \frac{\sum_n P(m|\mathbf{x}_n)(\mathbf{x}_n - \hat{\mu}_m)(\mathbf{x}_n - \hat{\mu}_m)^T}{\sum_n P(m|\mathbf{x}_n)} \\ &= \frac{\sum_n P(m|\mathbf{x}_n)(\mathbf{x}_n - \hat{\mu}_m)(\mathbf{x}_n - \hat{\mu}_m)^T}{N_m^*} \\ \hat{P}(m) &= \frac{1}{N} \sum_n P(m|\mathbf{x}_n) = \frac{N_m^*}{N}\end{aligned}$$

Example 1 fit using a GMM

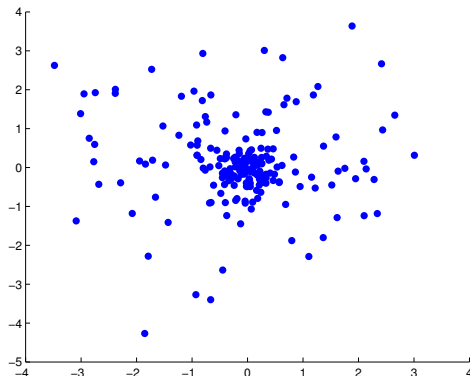


Example 1 fit using a GMM



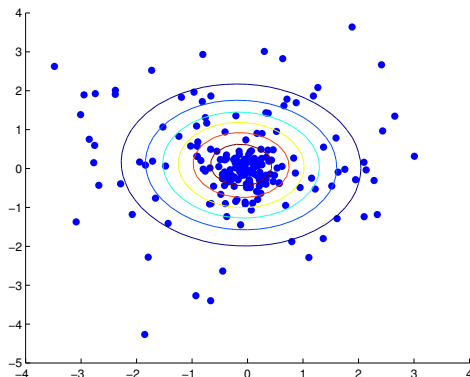
Fitted with a two component GMM using EM

Peakily distributed data (Example 2)



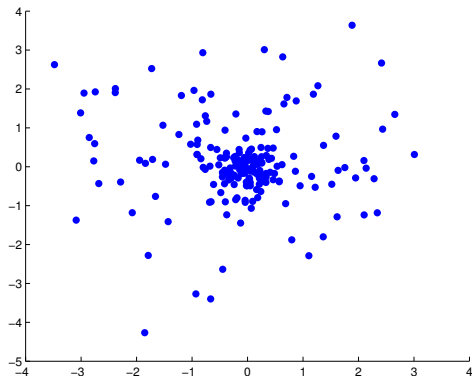
$$\mu_1 = \mu_2 = [0 \quad 0]^T \quad \Sigma_1 = 0.1I \quad \Sigma_2 = 2I$$

Example 2 fit by a Gaussian

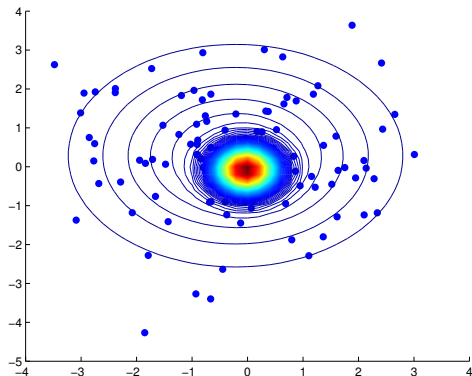


$$\mu_1 = \mu_2 = [0 \quad 0]^T \quad \Sigma_1 = 0.1I \quad \Sigma_2 = 2I$$

Example 2 fit by a GMM

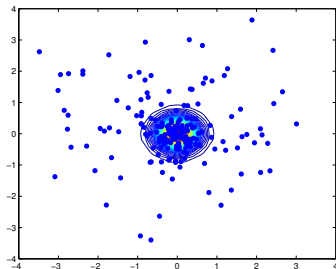


Example 2 fit by a GMM

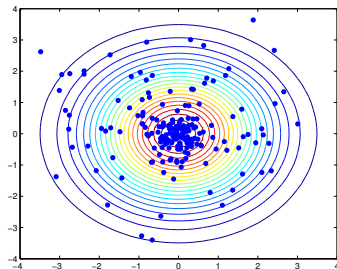


Fitted with a two component GMM using EM

Example 2: component Gaussians



$$P(x|m=1)$$



$$P(x|m=2)$$

- GMMs trained using the EM algorithm are able to self organise to fit a data set
- Individual components take responsibility for parts of the data set (probabilistically)
- Soft assignment to components not hard assignment — “soft clustering”
- GMMs scale very well, e.g.: large GMM-based speech recognition systems might have as many as 30,000 GMMs, each with 32 components: sometimes 1 million Gaussian components!! And the parameters all estimated from (a lot of) data by EM

HMMs with Gaussian observation probabilities

We can use a Gaussian distribution to model the observation probability:

$$b_j(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

We need to estimate parameters $\hat{\boldsymbol{\mu}}_j$, $\hat{\boldsymbol{\Sigma}}_j$ for each state j . Use the EM algorithm to weight each sample \mathbf{x}_t by the occupation probability $\gamma_j(t)$:

$$\hat{\boldsymbol{\mu}}_j = \frac{\sum_{t=1}^T \gamma_j(t) \mathbf{x}_t}{\sum_{t=1}^T \gamma_j(t)}$$

And likewise for the covariance matrices:

$$\hat{\boldsymbol{\Sigma}}_j = \frac{\sum_{t=1}^T \gamma_j(t) (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_t - \hat{\boldsymbol{\mu}}_j)^T}{\sum_{t=1}^T \gamma_j(t)}$$

Extension to Gaussian mixture model (GMM)

- The assumption of a Gaussian distribution at each state is very strong; in practice the acoustic feature vectors associated with a state may be strongly non-Gaussian
- In this case an M -component Gaussian mixture model is an appropriate density function:

$$b_j(\mathbf{x}) = p(\mathbf{x} | q = j) = \sum_{m=1}^M c_{jm} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm})$$

Given enough components, this family of functions can model any distribution.

- Train using the EM algorithm again, in which the component occupation probabilities are estimated along with the state occupation probabilities in the E-step

EM training of HMM/GMM

- Rather than estimating the state-time alignment, we estimate the component/state-time alignment, and component-state occupation probabilities $\gamma_{jm}(t)$: the probability of occupying mixture component m of state j at time t .

($\xi_{tm}(j)$ in Jurafsky and Martin's SLP)

- Re-estimate the parameters of component m of state j as follows

$$\hat{\mu}_{jm} = \frac{\sum_{t=1}^T \gamma_{jm}(t) \mathbf{x}_t}{\sum_{t=1}^T \gamma_{jm}(t)}$$
$$\hat{\Sigma}_{jm} = \frac{\sum_{t=1}^T \gamma_{jm}(t) (\mathbf{x}_t - \hat{\mu}_{jm})(\mathbf{x}_t - \hat{\mu}_{jm})^T}{\sum_{t=1}^T \gamma_{jm}(t)}$$

- The mixture coefficients are re-estimated in a similar way to transition probabilities:

$$\hat{c}_{jm} = \frac{\sum_{t=1}^T \gamma_{jm}(t)}{\sum_{m'=1}^M \sum_{t=1}^T \gamma_{jm'}(t)}$$

Doing the computation

- The forward, backward and Viterbi recursions result in a long sequence of probabilities being multiplied
- This can cause floating point *underflow* problems
- In practice computations are performed in the log domain (in which multiplies become adds)
- Working in the log domain also avoids needing to perform the exponentiation when computing Gaussians

- * Renals and Hain (2010). “Speech Recognition”, *Computational Linguistics and Natural Language Processing Handbook*, Clark, Fox and Lappin (eds.), Blackwells: section 2.2.