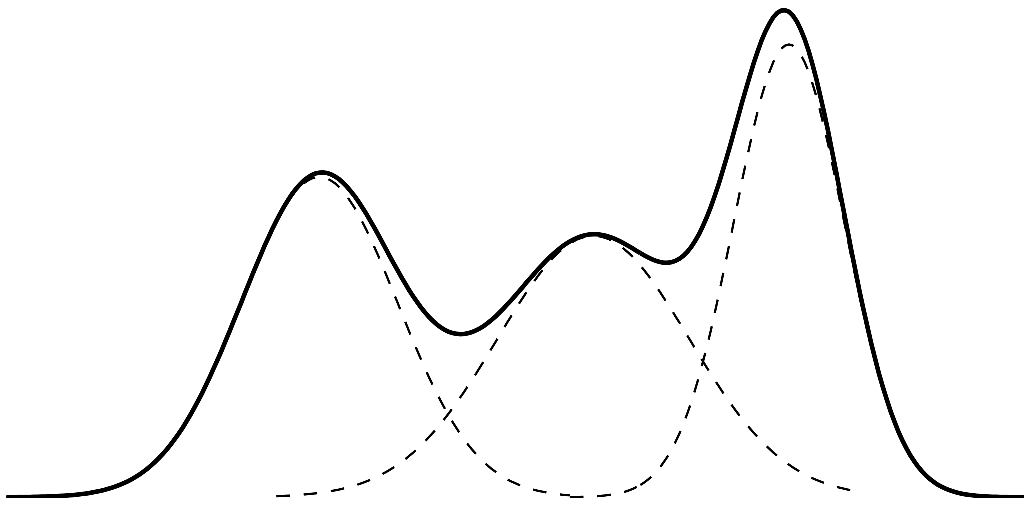
TW414 / PR813 Lecture 6
Gaussian Mixture Models



Ludwig Schwardt

University of Stellenbosch

7 April 2003

1. The Gaussian mixture model (GMM)

- GMM is "VQ on steroids" $\Longrightarrow$ each cluster has not only mean but also associated covariance matrix
- The GMM improves on VQ by being a true pdf $\Longrightarrow$ can therefore be slotted into Bayesian classifier
- A GMM is a ***mixture pdf*** — a linear combination of K Gaussian pdfs, or ***components***, given by

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} p(\boldsymbol{x}|k)P(k),$$

  where $P(k)$ is ***mixture weight*** subject to constraints

$$0 \leq P(k) \leq 1 \quad \textbf{and} \quad \sum_{k=1}^{K} P(k) = 1,$$

  and $p(\boldsymbol{x}|k)$ is height of kth component pdf at vector $\boldsymbol{x}$
- Mixture weight $P(k)$ has the form of an ***a priori*** probability $\Longrightarrow$ relative importance of each component in mixture pdf
- GMM is trained via EM algorithm, similar to K-means
- Component Gaussians can have full, diagonal or spherical covariance matrices $\Longrightarrow$ allows number of parameters to be tuned to suit the size of training data set
- A GMM can approximate any continuous density well, given enough components and suitable parameter values
- GMMs improve on standard Gaussians by allowing ***asymmetry*** and ***multimodality***, at cost of extra parameters

2. Training a GMM via EM

1. **Start with data set $\boldsymbol{X}$ of N feature vectors $\boldsymbol{x}_n, n = 1, \ldots, N$, initial set of K Gaussian component pdfs $\mathcal{N}_k \triangleq \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and K mixture weights $P(k),\ k = 1, \ldots, K$**
2. ***E-step:* Determine *responsibility* $P(k|\boldsymbol{x}_n)$ of each component pdf $\mathcal{N}_k$ for each training data point $\boldsymbol{x}_n$ as**

$$p_{kn} \triangleq P(k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|k)P(k)}{p(\boldsymbol{x}_n)},$$

   with GMM likelihood $p(\boldsymbol{x}_n) = \sum_{k=1}^{K} p(\boldsymbol{x}_n|k)P(k)$

3. ***M-step:* Re-estimate component pdfs and weights, based on data and responsibilities (compare Lecture 3):**

$$\begin{aligned}
\hat{P}(k) &= \frac{1}{N}\sum_{n=1}^{N} p_{kn}, \qquad \hat{\boldsymbol{\mu}}_k = \frac{\sum_n p_{kn}\boldsymbol{x}_n}{\sum_n p_{kn}}, \\
\hat{\boldsymbol{\Sigma}}_k &= \frac{\sum_n p_{kn}(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)^T}{\sum_n p_{kn}} \quad \textbf{(full cov)} \\
\hat{\sigma}_{ik}^2 &= \frac{\sum_n p_{kn}(x_{in} - \hat{\mu}_{ik})^2}{\sum_n p_{kn}}, \quad i = 1, \ldots, D \quad \textbf{(diag cov)} \\
\hat{\sigma}_k^2 &= \frac{\sum_n p_{kn}\left\|\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k\right\|^2}{D\sum_n p_{kn}} \quad \textbf{(spherical cov)}
\end{aligned}$$

4. **Repeat steps 2 to 3 until GMM likelihood $p(\boldsymbol{X}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n)$ of entire data set doesn't change appreciably, or limit on number of iterations is reached**

3. Using log likelihoods

- **Likelihoods are frequently too small to be directly represented as floating-point numbers $\Longrightarrow$ this *numerical underflow* solved by using log likelihoods instead**
- **This works great when likelihoods are multiplied $\Longrightarrow$ product becomes addition of log values**
- **When likelihoods are *added*, this poses a problem, since it cannot be done in log domain, and direct conversion back to linear domain again introduces underflow**
- **Standard "`logsumexp`" trick to calculate $\log(x_1 + \cdots + x_N)$ from log values $\log x_1$ to $\log x_N$ is to divide by largest term x_m and convert the scaled terms to linear domain instead:**

$$\mathrm{LSE}\left[\log x.\right] \triangleq \log\left(\sum_{n=1}^{N} x_n\right) = \log x_m + \log\left(\sum_{n=1}^{N} \mathrm{e}^{\log x_n - \log x_m}\right)$$

- **This changes calculation of GMM log likelihood to**

$$\log p(\boldsymbol{x}) = \mathrm{LSE}\left[\log p(\boldsymbol{x}|\cdot) + \log P(\cdot)\right]$$

- **Similarly, E-step in GMM training becomes**

$$P(k|\boldsymbol{x}_n) = \exp\left[\log p(\boldsymbol{x}_n|k) + \log P(k) - \log p(\boldsymbol{x}_n)\right]$$

  (responsibilities need to be in linear domain. . .), and log likelihood of entire data set is

$$\log p(\boldsymbol{X}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n)$$

4. Notes on calculation

- Good initialisation is crucial for good results, since EM algorithm only guarantees local optimum
- Standard approach initialises component means via VQ (e.g. binary split followed by K-means), chooses component covariance matrices as identity matrices (i.e. unity variance) and sets all mixture weights equal ($P(k) = 1/K$)
- Useful test to see if log likelihood $L = \log p(\boldsymbol{X})$ stabilised is to check relative increase in log likelihood from last iteration and stop if

$$\Delta L = \frac{L_{\text{curr}} - L_{\text{prev}}}{|L_{\text{prev}}|} < 5 \times 10^{-4}$$

- Algorithm should converge before about 100 iterations
- If K is large, some of the component pdfs may become ***degenerate*** during training (similar to K-means case) $\Longrightarrow$ too few data points are assigned to component, causing covariance matrix to become singular or ill-conditioned
- Useful solution to this problem is to discard offending component pdfs altogether, thereby reducing $K \Longrightarrow$ this can compensate for choosing K too high

5. GMM vs. K-means

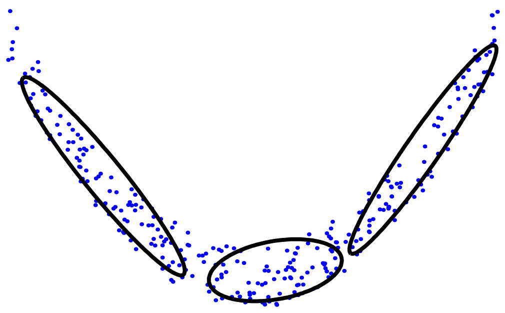
- GMM based on *probabilities*, while VQ based on *distances*
- GMM can be seen as "soft" form of VQ:
  - In K-means E-step, data points are assigned to nearest cluster $\Longrightarrow$ cluster membership is either 0% or 100%
  - In contrast, GMM responsibility $P(k|\boldsymbol{x}_n)$ plays rôle of ***soft*** cluster membership $\Longrightarrow$ since $\sum_k P(k|\boldsymbol{x}_n) = 1$, data point $\boldsymbol{x}_n$ belongs $P(k|\boldsymbol{x}_n) \times 100\%$ to cluster k, and each point contributes to some extent to each cluster
- "Hidden" information in K-means is cluster membership, while in GMM it is responsibilities $\Longrightarrow$ EM algorithm iterates between model parameters and responsibilities
- Comparing re-estimation formulae of GMM with standard Gaussian MLE equations, we see that terms with form

  $$\frac{1}{N}\sum_{n=1}^{N} f(n) \quad \text{are replaced by} \quad \frac{1}{\sum_n p_{kn}}\sum_{n=1}^{N} p_{kn} f(n)$$

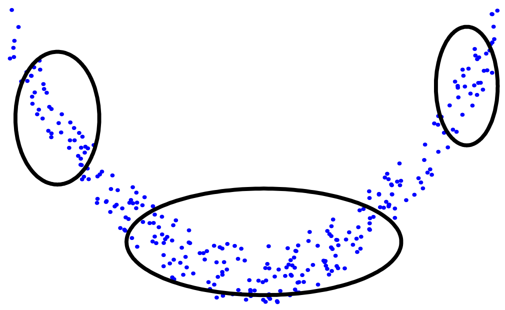
  $\Longrightarrow$ GMM therefore weights contribution of each point with responsibility (same answers if $p_{kn} = 1/K$)
- K-means is special limiting case of GMM with equal mixture weights, equal spherical component covariance matrices (i.e. $\Sigma_k = \sigma^2 \boldsymbol{I}$) and variance σ^2 approaching zero

6. Effect of covariance type

- **Full covariance GMM fits data best $\Longrightarrow$ costly in high-dimensional feature spaces though (consider PCA/LDA)**



- **Diagonal covariance GMM good compromise between quality and model size $\Longrightarrow$ good bang for parameter buck**



- **Spherical covariance GMM needs many components to cover data, especially in high-dimensional feature spaces**