

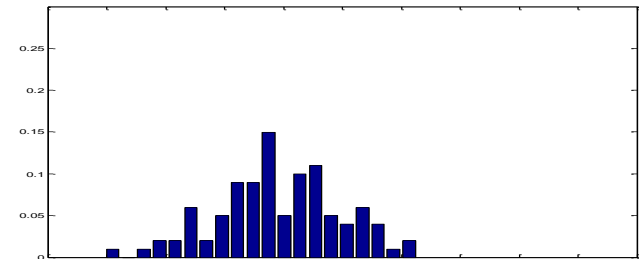
Unsupervised Learning

Density Estimation

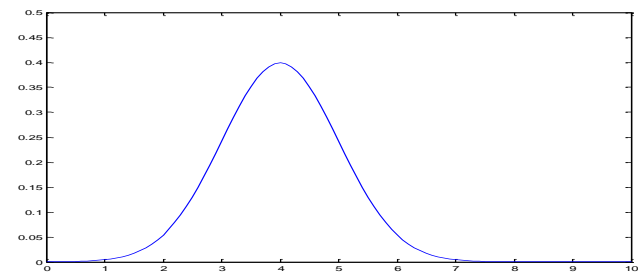
Ioannis Patras

Density Estimation

Data histogram



Learned density

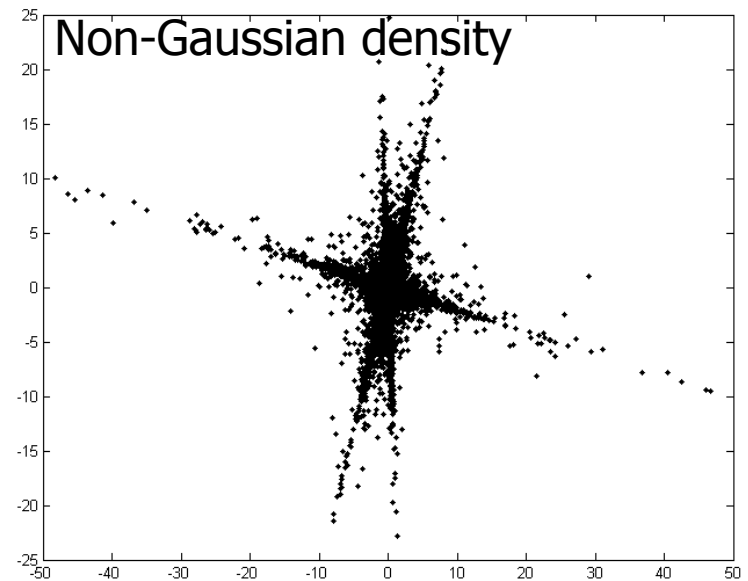


Model the density of the probability from which \mathbf{x} is drawn. $\mathbf{x} \sim p(\mathbf{x}; \mathbf{w})$, where \mathbf{w} represents an unknown parameter vector that needs to be learned.

Density estimation

Not all data is well represented by Gaussian models...

Therefore need a possible way of representing more general densities



Mixture Models

Idea of mixture model: Assume there are some unobservable discrete variables (also called latent variables) underlying the observed data (c.f. classification)

Fish example. We observe 'fish lightness', x , and have two models, one for Salmon and one for Sea Bass. If we are not interested in classifying the fish (but only in the lightness variability) → Total probability model:

$$P(x) = P(\text{Salmon}) P(x/\text{Salmon}) + P(\text{SeaBass}) P(x/\text{SeaBass})$$

Mixture Models

Idea of mixture model: Assume there are some unobservable discrete variables (also called latent variables) underlying the observed data (c.f. classification)

The general *mixture model* with K different components has the following form

$$P(x) = \sum_k P(c_k)P(x / c_k) = \sum_k P(x, c_k)$$

Remember the classes c_k are not observed (unsupervised learning).

Mixture model example:

Mixtures of Gaussians (MoGs)

One mixture model is the mixture of Gaussians (MoGs) which has the general form:

$$\begin{aligned} P(x) &= \sum_k P(c_k) P(x | c_k) \\ &= \sum_k P(c_k) N(\mu_k, \Sigma_k) \\ &= \sum_{k=1}^K P(c_k) \frac{1}{|\Sigma_k|^{d/2}} \exp\left(-\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k)\right) \end{aligned}$$

Learning mixture models I

How do we estimate mixture models? Here we will look at ML. Let θ represent the parameters of interest.

Let $p(x | \theta) = \sum_{k=1}^K P(c_k) P(x | c_k, \theta_k)$ so that:

$$\begin{aligned} E &= -\ln p(\chi | \theta) = \\ &= -\sum_{n=1}^N \ln p(x_n | \theta) \quad (\text{assuming independent observations}) \end{aligned}$$

Solve by setting the partial derivatives to zero

$$\frac{\partial E}{\partial \theta} = -\sum_{n=1}^N \frac{\partial \ln p(x_n | \theta)}{\partial \theta} = -\sum_{n=1}^N \frac{\partial p(x_n | \theta) / \partial \theta}{p(x_n | \theta)} = 0 \quad (\text{since } \frac{\partial}{\partial \theta} \ln f(\theta) = \frac{\partial f(\theta) / \partial \theta}{f(\theta)})$$

differentiating w.r.t parameters in the k th mixture:

$$\frac{\partial E}{\partial \theta_k} = -\sum_{n=1}^N \frac{p(c_k)}{p(x_n | \theta)} \frac{\partial}{\partial \theta_k} p(x_n | c_k, \theta_k) = 0$$

(got by noting that $\frac{\partial}{\partial \theta_k} p(x_n | \theta) = \frac{\partial}{\partial \theta_k} \sum_j p(c_j) p(x_n | c_j, \theta_j) = \frac{\partial}{\partial \theta_k} p(c_k) p(x_n | c_k, \theta_k)$)

Learning mixture models II

Applying $\partial(\ln f(\theta))/\partial\theta = \frac{\partial f(\theta)/\partial\theta}{f(\theta)}$ again gives:

$$\begin{aligned}\frac{\partial E}{\partial\theta_k} &= -\sum_{n=1}^N \frac{p(c_k)p(x_n | c_k, \theta)}{p(x_n | \theta)} \frac{\partial}{\partial\theta_k} \ln p(x_n | c_k, \theta) \\ &= -\sum_{n=1}^N p(c_k | x_n, \theta) \frac{\partial}{\partial\theta_k} \ln p(x_n | c_k, \theta)\end{aligned}$$

Applying Bayes rule $\frac{p(c_k | \theta)p(x_n | c_k, \theta)}{p(x_n | \theta)} = p(c_k | x_n, \theta)$

and noting that $p(c_k) = p(c_k | \theta)$

This is a weighted version of what we would get if we knew the classes

That is: we weight the contribution of x_n with the probability that it belongs to class c_k :

Parameter estimation (cont.)

$$\sum_{n=1}^N p(c_k / x_n, \theta) \frac{\partial}{\partial \theta_k} \ln p(x_n | c_k, \theta_k) = 0$$

Ignoring the $p(c_k / x_n, \theta)$ term we have the standard ML solution *given* component c_k .

The $p(c_k / x_n, \theta)$ term is the probability that the point x_n belongs to the c_k component model.

Can we use this to compute the ML solution for the mixture?

EM iterative update

Although $p(c_k / x_n, \theta)$ is not independent of θ_k we can treat it as such and generate the following iterative algorithm:

Expectation - Maximization (beginning with initial values $\hat{\theta}^{(0)}$)

E – step

calculate: $q_{n,k} = p(c_k | x_n, \hat{\theta}^{(i)})$ for all k and n

M-step

maximise the weighted log likelihood:

$$\hat{\theta}_k^{(i+1)} = \operatorname{argmax}_{\theta} \sum_n q_{n,k} \ln p(x_n | c_k, \theta_k)$$

for all k

A converged solution is a stationary point (e.g. local maximum) of the likelihood

EM algorithm for MoGs I

We can now easily apply the EM algorithm to estimate MoG parameters.

Let $w_k^{(i)}$ represent the i th estimate for $p(c_k)$:

•E-step : Here we calculate $q_{n,k} = p(c_k | x_n, \theta^{(i)})$ by using

$$\begin{aligned} p(c_k | x_n, \theta^{(i)}) &= p(x_n | c_k, \theta^{(i)}) p(c_k) / p(x_n | \theta^{(i)}) \\ &= p(x_n | c_k, \theta^{(i)}) p(c_k) / \sum_j p(x_n | c_j, \theta^{(i)}) p(c_j) \end{aligned}$$

$$\text{Where } p(x_n | c_k, \theta^{(i)}) = \frac{1}{\Sigma_k^{d/2}} \exp\left(-\frac{1}{2} (x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k)\right)$$

EM algorithm for MoGs II

•M-step :

$$1. \mu_k^{(i+1)} = \sum_n q_{n,k} x_n / \sum_n q_{n,k} \quad (\text{weighted mean})$$

$$2. \Sigma_k^{(i+1)} = \frac{\sum_n q_{n,k} (x_n - \mu_k^{(i+1)})(x_n - \mu_k^{(i+1)})^T}{\sum_n q_{n,k}} \quad (\text{weighted covariance})$$

It is also possible to show that we can estimate the priors $p(c_k)$ for each class c_k using the following update:

$$3. w_k^{(i)} = \frac{1}{N} \sum_n q_{n,k}$$

All the variables can be updated as weighted averages!

EM iterative update (E-step)

Expectation - Maximization (beginning with initial values $\hat{\theta}^{(0)}$)

E – step calculate: $q_{n,k} = p(c_k | x_n, \hat{\theta}^{(i)})$ for all k and n

M-step maximise the weighted log likelihood:

$$\hat{\theta}_k^{(i+1)} = \operatorname{argmax}_{\theta} \sum_n q_{n,k} \ln p(x_n | c_k, \theta_k) \quad \text{for all } k$$

EM iterative update (E-step)

Expectation - Maximization (beginning with initial values $\hat{\theta}^{(0)}$)

E – step calculate: $q_{n,k} = p(c_k | x_n, \hat{\theta}^{(i)})$ for all k and n

M-step maximise the weighted log likelihood:

$$\hat{\theta}_k^{(i+1)} = \operatorname{argmax}_{\theta} \sum_n q_{n,k} \ln p(x_n | c_k, \theta_k) \quad \text{for all } k$$

EM iterative update (M-step)

Expectation - Maximization (beginning with initial values $\hat{\theta}^{(0)}$)

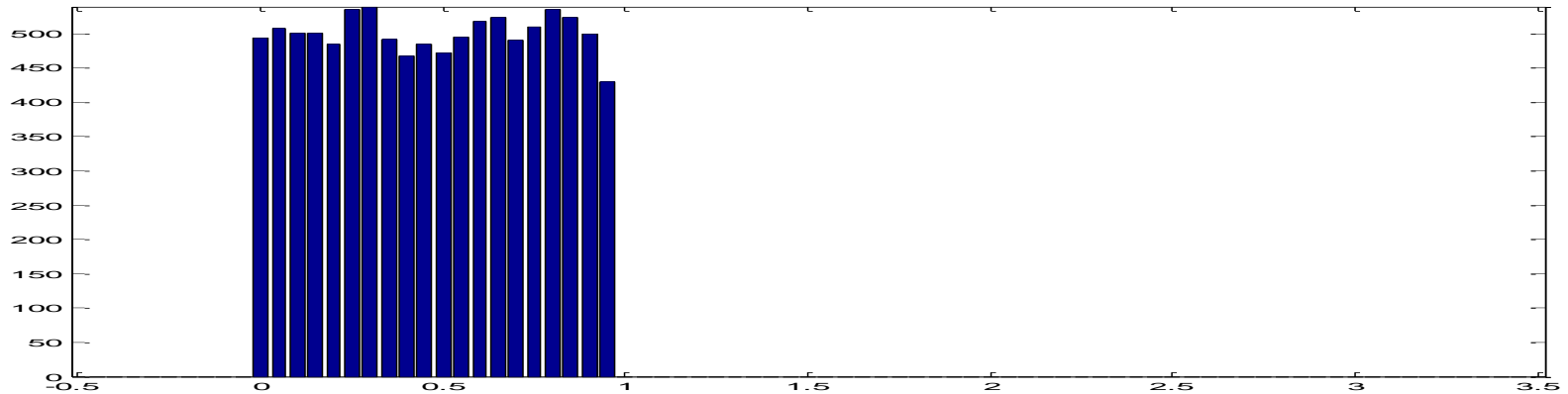
E – step calculate: $q_{n,k} = p(c_k | x_n, \hat{\theta}^{(i)})$ for all k and n

M-step maximise the weighted log likelihood:

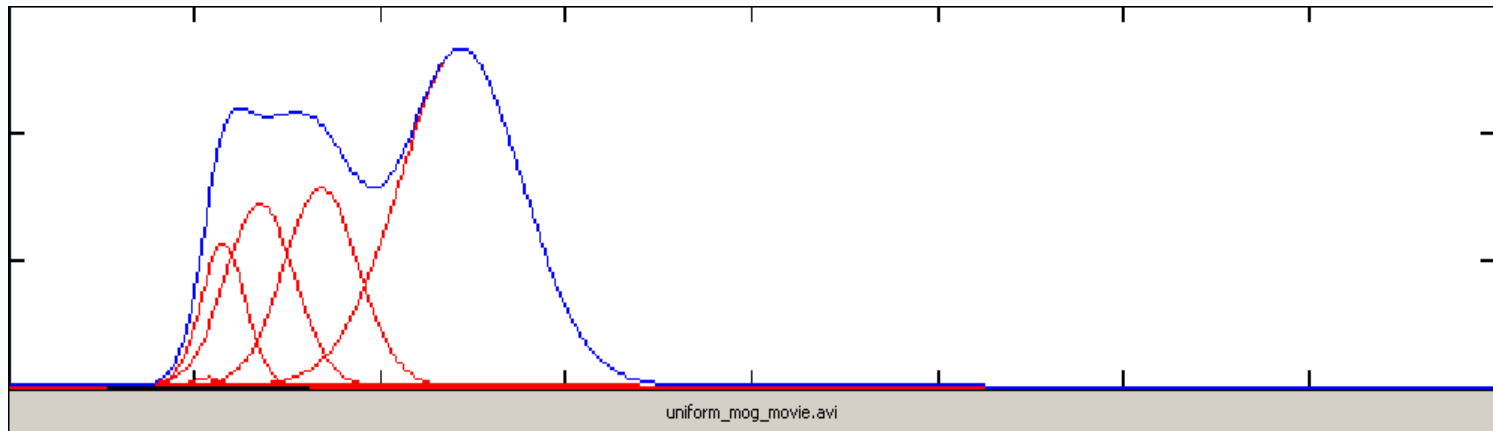
$$\hat{\theta}_k^{(i+1)} = \operatorname{argmax}_{\theta} \sum_n q_{n,k} \ln p(x_n | c_k, \theta_k) \quad \text{for all } k$$

$$\bullet \text{M-step : } \mu_k^{(i+1)} = \sum_n q_{n,k} x_n / \sum_n q_{n,k} \quad (\text{weighted mean})$$

EM algorithm for MoG example



This demonstrates 100 EM iterations of a 5 Gaussian MoG



Mixtures of Gaussians (MoGs)

MoGs are very flexible and can be used to approximate any density:

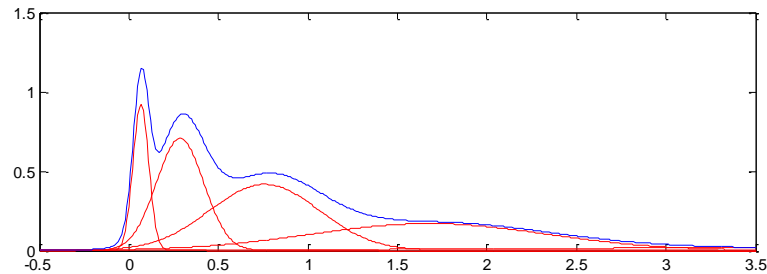
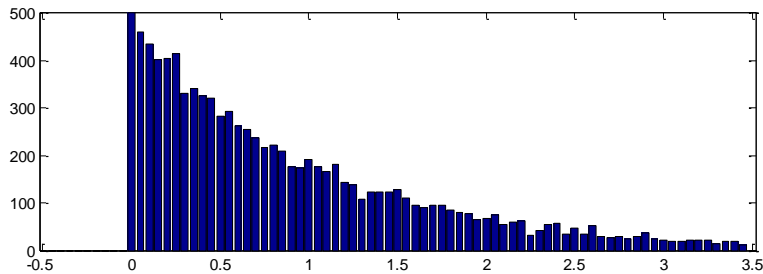
Theorem: *Any $p(x)$ can be approximated as closely as desired by a Gaussian mixture: i.e. there is an order K and priors $p(c_k)$ for which*

*For any $\varepsilon > 0$. (Anderson and Moore, *Optimal Filtering* 1979)*

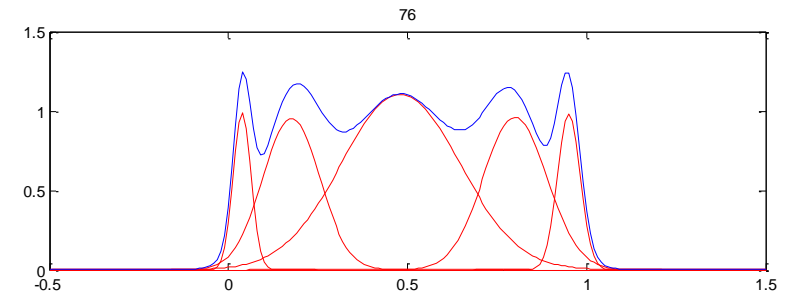
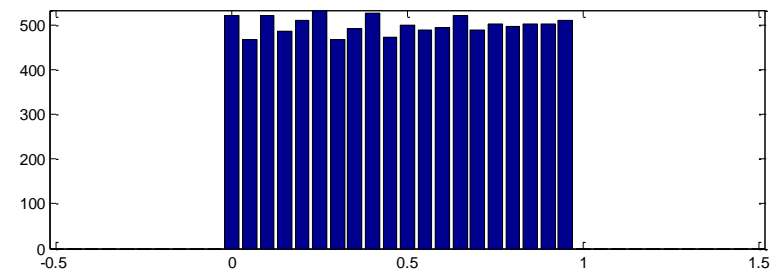
$$\int_x \left| P(x) - P_{\text{mog}}(x) \right| dx < \varepsilon$$

Examples of MoGs

Here are a couple of MoG approximations using 5 Gaussians:



Exponential distribution



Uniform distribution

Notice in these two cases the individual Gaussians have no special meaning.

Convergence issues

- Number of Gaussians is assumed to be known. Determining the correct number in an unsupervised way is an ill-posed problem.
- Converges to a LOCAL minimum. Run with several random initialisations. Select the one that maximizes the data likelihood.

$$P(x_n | \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K) \propto \sum_{k=1}^K \frac{P(c_k)}{\det(\Sigma_k)^{1/2}} \exp\left(-\frac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1}(x_n - \mu_k)\right)$$

Convergence issues

Overfitting. MoGs can potentially exhibit a very interesting form of overfitting – **Singular points**

Recall the likelihood for x_n is:

$$P(x_n | \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K) \propto \sum_{k=1}^K \frac{P(c_k)}{\det(\Sigma_k)^{1/2}} \exp\left(-\frac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1}(x_n - \mu_k)\right)$$

Now suppose $x_n = \mu_k$ for some k and $\Sigma_k \rightarrow 0$. Then

$$P(x_n | \mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K) \approx \frac{P(c_k)}{\det(\Sigma_k)^{1/2}} \exp(0) \rightarrow \infty$$

So the likelihood $\rightarrow \infty$. These singular points ALWAYS exist. They can be avoided by constraining the covariances always above some minimum value.

Overfitting in ML density estimation

$$p(x | \mu, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

2. Regularise the covariance matrix by adding a diagonal matrix $\Sigma_0 = \sigma_0^2 I$

$$\hat{\Sigma} = \sum_n \left(x_n - \mu \right) \left(x_n - \mu \right)^T + \sigma_0^2 I =$$

Leads to “thicker” Gaussian. Equivalent to adding zero mean Gaussian noise (with $\Sigma_0 = \sigma_0^2 I$) to the original dataset.



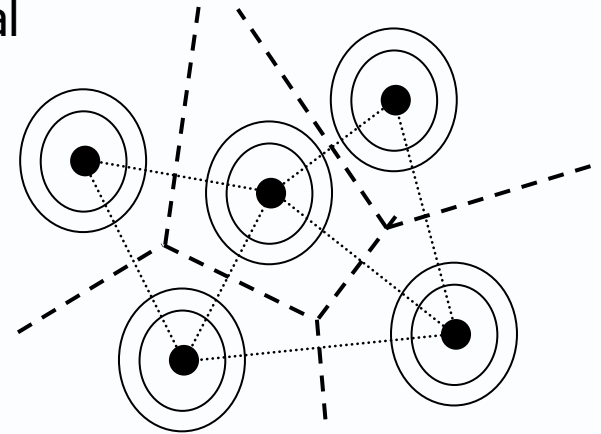
K - means MoG link

We can make the K-means/MoG link more explicit. Recall that Isotropic Gaussians (covariance = $\sigma^2 \mathbf{I}$) with equal weights (class priors) \rightarrow templates model

E-step:

Step 1 = full classification (as opposed to weighting with probabilities) $\rightarrow \mathbf{P}_{n,k} = \mathbf{0}$ or $\mathbf{1}$.

This is only true if $\sigma^2 \rightarrow \mathbf{0}$



M-step:

The only parameters to be updated are the means μ_k . So this is identical to the MoG M-step.

So K-means \sim MoG with isotropic covariances $\Sigma_k \rightarrow \mathbf{0}$ and equal class priors.

Summary

- o Limitations of Gaussian models for unsupervised learning
- o Introduced mixture modelling
 - Latent variables
 - Looked at the special case of the MoGs
 - Parameter estimation via EM
 - Convergence issues