Gaussian Mixture Models

A **mixture model** simply means that the pdf for a random variable/vector we are modeling is (convex) weighted sum of different "component" pdfs. That is,

$$f_X(\boldsymbol{x}) = \sum_{q=1}^{Q} \beta_q \phi_q(\boldsymbol{x}),$$

where the ϕ_q are themselves pdfs,

$$\phi_q(\boldsymbol{x}) \geq 0, \quad \text{and} \quad \int_{\boldsymbol{x}\in\mathbb{R}^D} \phi_q(\boldsymbol{x}) \ \mathrm{d}\boldsymbol{x} = 1,$$

and hence the weights must obey

$$\beta_q \geq 0, \quad \text{and} \quad \sum_{q=1}^{Q} \beta_q = 1.$$

There are two lenses through which we might view mixture models. The first is simply a way to approximate pdfs with superpositions of a known form (much like approximating a function with a superposition of basis functions). The other is to interpret the random variable as being controlled by a latent "state" — there are Q different modes the random variable might act in, each with a different probability β_q, with the ϕ_q specifying the behavior in each mode.

The above defines a clear generative model. To generate a realization of X, we first generate a state $q \in \{1, \ldots, Q\}$, then draw X using the pdf $\phi_q(\boldsymbol{x})$ of the selected state.

Here are a few examples where a mixture model is a good way to describe the data.

Demodulation in digital communications

Suppose that a transmitter sends $+1$ with probability $1/2$ and -1 with probability $1/2$, and that this number is corrupted by adding a noise variable $Z \sim \text{Normal}(0, \sigma^2)$. Then the receiver observes a random variable that is the mixture of two Gaussians.

This is a model that is commonly used in digital communications, only the mixtures could have many more components — usually instead of sending a bit at a time, the transmitter will send one of 8 or 16 or even 64 symbols.

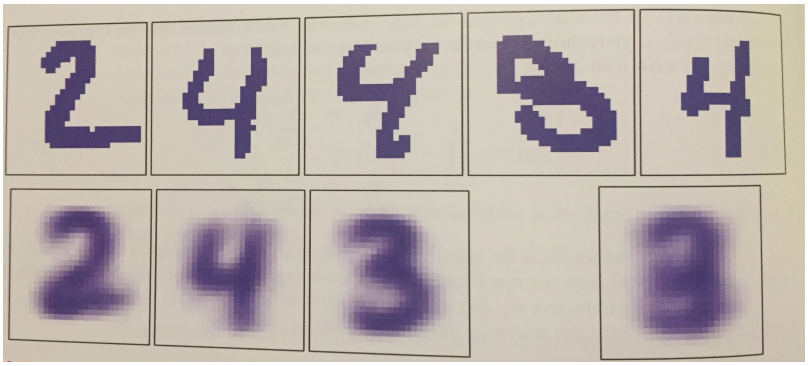
Pixel clustering in images

The pixel values in different "segments" (regions corresponding to different objects) in an image might be interpreted as random variables clustered around different means

Example from C. Bishop, *Pattern Recognition and Machine Learning*, Chapter 9.

Binary mixtures

If we are analyzing a binary image that we are expecting to be one of 10 digits (say), then each pixel is "turned on" with a different probability for every digit.



Example from C. Bishop, *Pattern Recognition and Machine Learning*, Chapter 9.

Gaussian mixture models

As the name suggests, a *Gaussian mixture model* for a random variable/vector X simply means that its pdf is a convex combination of normal pdfs with different means and different covariance matrices:

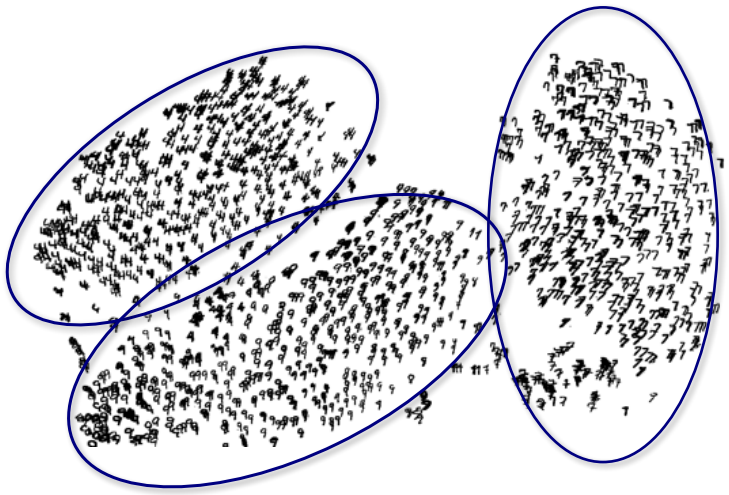
$$f_X(\boldsymbol{x}) = \sum_{q=1}^{Q} \beta_q \phi_q(\boldsymbol{x}),$$

with

$$\phi_q(\boldsymbol{x}) = \phi(\boldsymbol{x}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) = \frac{1}{(2\pi)^{D/2}\sqrt{\det \boldsymbol{\Sigma}_q}} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_q)^{\mathrm{T}}\boldsymbol{\Sigma}_q^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_q)}{2}\right) \tag{1}$$

for $\boldsymbol{\mu}_q \in \mathbb{R}^D$ and $\boldsymbol{\Sigma}_d \in \mathcal{S}_{++}^D$.

One geometric way we might interpret a GMM is that it is generating "features" in different "classes" indexed by q, and the data points in those classes are clustered in ellipsoids. This is pretty decent model in a surprising number of situations ... here is a simple example of 3 digits from the MNIST data set projected into $\mathbb{R}^2$:



It is **very** important to realize that mixture distributions are different than taking convex combinations of the random variables themselves. For example, suppose that

$$X_1 \sim \text{Normal}(-1, 1), \quad X_2 \sim \text{Normal}(1, 1).$$

Sketch $f_{X_1}(x)$ and $f_{X_2}(x)$:

Now sketch the mixture pdf $\frac{1}{2}f_{X_1}(x) + \frac{1}{2}f_{X_2}(x)$:

Now sketch the pdf of $\frac{1}{2}X_1 + \frac{1}{2}X_2$:

Estimating the parameters of a GMM

Given a series of (independent) observations $X_1 = \boldsymbol{x}_1, \ldots, X_N = \boldsymbol{x}_N$ of a Gaussian mixture, how do we estimate the parameters $\beta_1, \ldots, \beta_Q, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_Q$ and $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_Q$? Well, let's set up the MLE. With

$$\boldsymbol{\theta} = \{\beta_1, \ldots, \beta_Q, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_Q, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_Q\},$$

the likelihood function is

$$L(\boldsymbol{\theta}; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \prod_{n=1}^{N} \left(\sum_{q=1}^{Q} \beta_q \, \phi(\boldsymbol{x}_n; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) \right),$$

where $\phi(\boldsymbol{x}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$ is as in (1), and the log likelihood is

$$\ell(\boldsymbol{\theta}; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \sum_{n=1}^{N} \log \left(\sum_{q=1}^{Q} \beta_q \, \phi(\boldsymbol{x}_n; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) \right).$$

The problem, besides the sheer number of parameters that we are estimating, and besides the fact that there is no closed form minimizer, is that this is a highly non-concave function of the $\{\beta_q\}$, $\{\boldsymbol{\mu}_q\}$ and $\{\boldsymbol{\Sigma}_q\}$. So there are no computationally reasonable algorithms for minimizing the expression above that come with rock solid guarantees about finding the global max.

But what if we knew the state variables for each of the X_n? We know that a method for generating X_n proceeds by drawing a state, which we will call $S_n \in \{1, \ldots, Q\}$, and then drawing a Gaussian random variable (still with unknown mean and variance) based on S_n. With these additional observations $S_1 = s_1, \ldots, S_N = s_n$, the log likelihood function is

$$\begin{aligned} \ell(\boldsymbol{\theta}; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, s_1, \ldots, s_N) &= \sum_{n=1}^{N} \log \left(\beta_{s_n} \phi(\boldsymbol{x}_n; \boldsymbol{\mu}_{s_n}, \boldsymbol{\Sigma}_{s_n}) \right) \\ &= \sum_{n=1}^{N} \log \beta_{s_n} + \sum_{n=1}^{N} \log \phi(\boldsymbol{x}_n; \boldsymbol{\mu}_{s_n}, \boldsymbol{\Sigma}_{s_n}). \end{aligned}$$

A simple calculation shows that the maximum of the above under the constraints that $\beta_q \geq 0$, $\sum_q \beta_q = 1$, and the $\boldsymbol{\Sigma}_q$ are symmetric positive definite is given by the following very intuitive expressions. Let $\mathcal{I}_q$ be the set of all n that has state q:

$$\mathcal{I}_q = \{n \ : \ s_n = q\}.$$

The the maximizer of $\ell(\boldsymbol{\theta}; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, s_1, \ldots, s_N)$ is

$$\begin{aligned}
\hat{\beta}_q &= \frac{|\mathcal{I}_q|}{N}, \\
\hat{\boldsymbol{\mu}}_q &= \frac{1}{|\mathcal{I}_q|} \sum_{n \in \mathcal{I}_q} \boldsymbol{x}_n, \\
\hat{\boldsymbol{\Sigma}}_q &= \frac{1}{|\mathcal{I}_q|} \sum_{n \in \mathcal{I}_q} (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_q)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_q)^{\mathrm{T}},
\end{aligned}$$

where $|\mathcal{I}_q|$ is the size of the set $\mathcal{I}_q$.

But alas, we do not get to observe the $S_1, \ldots, S_N$. So here is what we can do. We start with an initial guess of all the parameters $\boldsymbol{\theta}^{(0)}$. Then we compute a likelihood function that is **averaged** over all possible state realizations, where the average is computed using the variables in $\boldsymbol{\theta}^{(0)}$. That is, we form the function

$$\ell^{(0)}(\boldsymbol{\theta}) = \mathrm{E}[\ell(\boldsymbol{\theta}; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, S_1, \ldots, S_N); \boldsymbol{\theta}^{(0)}],$$

and then take

$$\boldsymbol{\theta}^{(1)} = \arg \max_{\boldsymbol{\theta} \in \mathcal{T}} \ell^{(0)}(\boldsymbol{\theta}). \tag{2}$$

The expectation in the first expression above is over the states $S_1, \ldots, S_N$ using the model given by $\boldsymbol{\theta}^{(0)}$ as is computed as follows:

$$\begin{aligned}
\mathrm{E}[\ell(\boldsymbol{\theta}; \boldsymbol{x}_1, \ldots, &\boldsymbol{x}_N, S_1, \ldots, S_N); \boldsymbol{\theta}^{(0)}] = \\
&\sum_{n=1}^{N} \sum_{q=1}^{Q} \gamma_{n,q}^{(0)} \left(\log \beta_q + \log \phi(\boldsymbol{x}_n; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) \right)
\end{aligned}$$

where

$$\begin{aligned}\gamma_{n,q}^{(0)} &= \mathrm{P}\left(S_n = q | X_n = \boldsymbol{x}_n; \boldsymbol{\theta}^{(0)}\right) \\ &= \frac{f_{X_n}(\boldsymbol{x}_n | S_n = q; \boldsymbol{\theta}^{(0)}) \, \mathrm{P}\left(S_n = q; \boldsymbol{\theta}^{(0)}\right)}{\sum_{q'=1}^{Q} f_{X_n}(\boldsymbol{x}_n | S_n = q'; \boldsymbol{\theta}^{(0)}) \, \mathrm{P}\left(S_n = q'; \boldsymbol{\theta}^{(0)}\right)} \\ &= \frac{\beta_q^{(0)} \phi(\boldsymbol{x}_n; \boldsymbol{\mu}_q^{(0)}, \boldsymbol{\Sigma}_q^{(0)})}{\sum_{q'=1}^{Q} \beta_{q'}^{(0)} \phi(\boldsymbol{x}_n; \boldsymbol{\mu}_{q'}^{(0)}, \boldsymbol{\Sigma}_{q'}^{(0)})}.\end{aligned}$$

With the $\gamma_{n,q}^{(0)}$, the maximizer in (2) is again straightforward. Instead of taking a hard partition, as we did when the states were known, we take a weighted average based on the $\gamma_{n,q}^{(0)}$, i.e.

$$\begin{aligned}\beta_q^{(1)} &= \frac{1}{N} \sum_{n=1}^{N} \gamma_{n,q}^{(0)}, \\ \boldsymbol{\mu}_q^{(1)} &= \frac{\sum_{n=1}^{N} \gamma_{n,q}^{(0)} \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma_{n,q}^{(0)}} \\ \boldsymbol{\Sigma}_q^{(1)} &= \frac{\sum_{n=1}^{N} \gamma_{n,q}^{(0)} (\boldsymbol{x}_n - \boldsymbol{\mu}_q^{(1)})(\boldsymbol{x}_n - \boldsymbol{\mu}_q^{(1)})^{\mathrm{T}}}{\sum_{n=1}^{N} \gamma_{n,q}^{(0)}}.\end{aligned}$$

We can then iterate, taking

$$\boldsymbol{\theta}^{(k)} = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}} \mathrm{E}[\ell(\boldsymbol{\theta}; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, S_1, \ldots, S_N); \boldsymbol{\theta}^{(k-1)}]$$

This is a special instance of what is known as the **Expectation Maximization** (EM) algorithm — the iterations above can be generalized to many different problems where there are latent variables that cause different behaviors in the random variables. It is a classical result that the EM will converge to a local maxima of the log

likelihood function, and in fact it monotonically ascends on the likelihood,

$$\ell(\boldsymbol{\theta}^{(k+1)}; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) \geq \ell(\boldsymbol{\theta}^{(k)}; \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$$

but there is no guarantee that it finds the global maximizer. Still, this algorithm is widely used.

EM algorithm for learning a GMM

Given initialization $\boldsymbol{\theta}^{(0)} = \{\beta_q^{(0)}, \boldsymbol{\mu}_q^{(0)}, \boldsymbol{\Sigma}_q^{(0)},\ q = 1, \ldots, Q\}$,

while not converged, $\ell(\boldsymbol{\theta}^{(k)}; \cdots) - \ell(\boldsymbol{\theta}^{(k-1)}; \cdots) < \delta$ **do**

$$\gamma_{n,q}^{(k)} = \frac{\beta_q^{(k)} \phi(\boldsymbol{x}_n; \boldsymbol{\mu}_q^{(k)}, \boldsymbol{\Sigma}_q^{(k)})}{\sum_{q'=1}^{Q} \beta_{q'}^{(k)} \phi(\boldsymbol{x}_n; \boldsymbol{\mu}_{q'}^{(k)}, \boldsymbol{\Sigma}_{q'}^{(k)})}, \text{ for } q = 1, \ldots, Q$$

$$\beta_q^{(k+1)} = \frac{1}{N} \sum_{n=1}^{N} \gamma_{n,q}^{(k)}, \text{ for } q = 1, \ldots, Q$$

$$\boldsymbol{\mu}_q^{(k+1)} = \frac{\sum_{n=1}^{N} \gamma_{n,q}^{(k)} \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma_{n,q}^{(k)}} \text{ for } q = 1, \ldots, Q$$

$$\boldsymbol{\Sigma}_q^{(k+1)} = \frac{\sum_{n=1}^{N} \gamma_{n,q}^{(k)} (\boldsymbol{x}_n - \boldsymbol{\mu}_q^{(k+1)})(\boldsymbol{x}_n - \boldsymbol{\mu}_q^{(k+1)})^{\mathrm{T}}}{\sum_{n=1}^{N} \gamma_{n,q}^{(k)}} \text{ for } q = 1, \ldots, Q$$

$k = k + 1$

end while