MML Gaussian Mixture Models

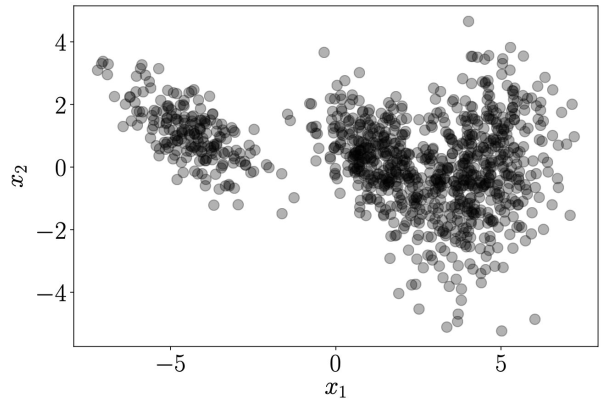
San Diego Machine Learning
Ryan Chesler

Overview

- We want to be able to represent our data compactly

- We have certain well established distributions but they might not well represent our data alone
    - Gaussian, Beta, Poisson, etc.
- The real world is messy and might have multiple overlapping distributions

- Using gaussian mixture models we can well describe these cases

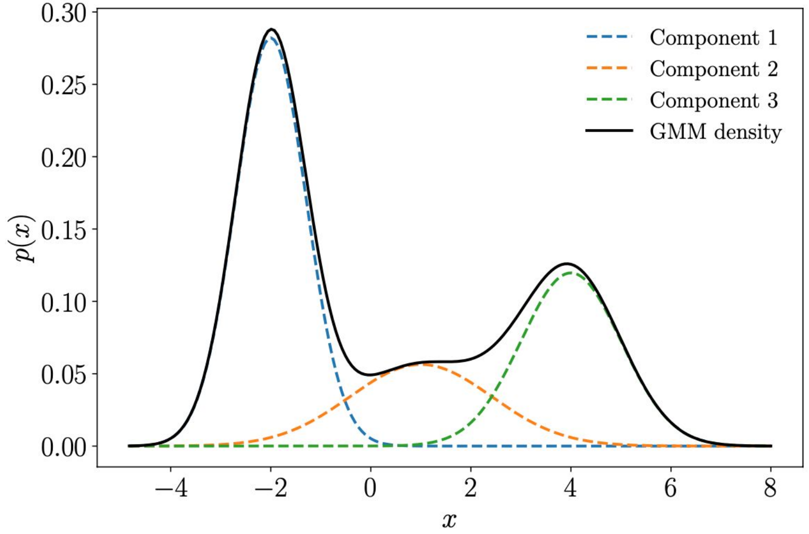


Gaussian Mixture Model

- pk - components/basic distributions
- πk - mixture weights
- K - number of clusters

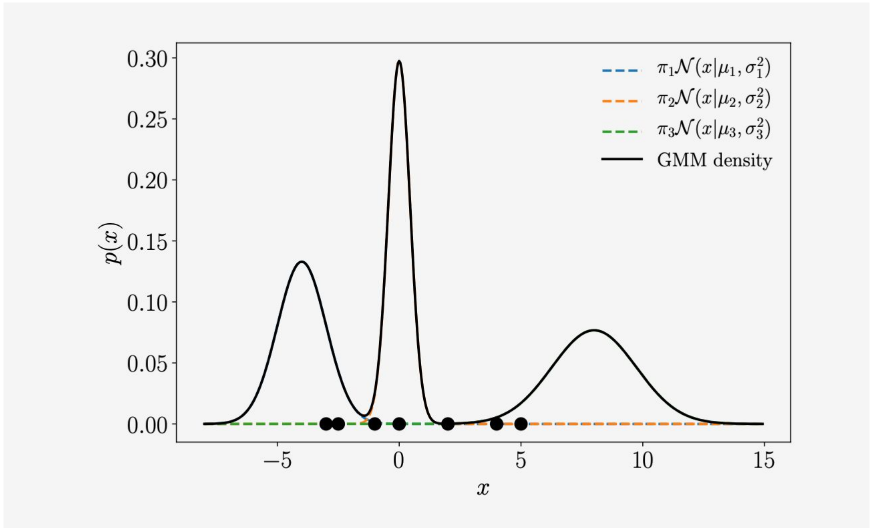
$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x})$$

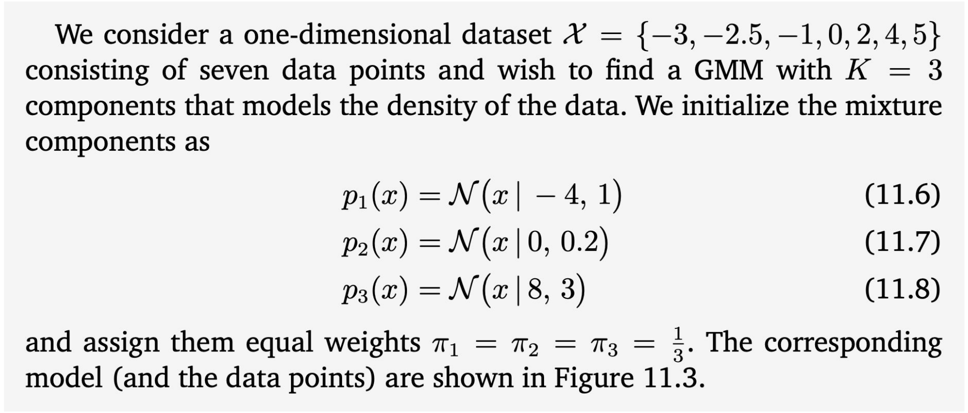
$$0 \leqslant \pi_k \leqslant 1, \quad \sum_{k=1}^{K} \pi_k = 1$$



$$p(x \mid \boldsymbol{\theta}) = 0.5\mathcal{N}\left(x \mid -2,\ \tfrac{1}{2}\right) + 0.2\mathcal{N}\left(x \mid 1,\ 2\right) + 0.3\mathcal{N}\left(x \mid 4,\ 1\right.$$

Example





We consider a one-dimensional dataset $\mathcal{X} = \{-3, -2.5, -1, 0, 2, 4, 5\}$ consisting of seven data points and wish to find a GMM with $K = 3$ components that models the density of the data. We initialize the mixture components as

$$p_1(x) = \mathcal{N}\big(x \,|\, -4,\ 1\big) \tag{11.6}$$

$$p_2(x) = \mathcal{N}\big(x \,|\, 0,\ 0.2\big) \tag{11.7}$$

$$p_3(x) = \mathcal{N}\big(x \,|\, 8,\ 3\big) \tag{11.8}$$

and assign them equal weights $\pi_1 = \pi_2 = \pi_3 = \frac{1}{3}$. The corresponding model (and the data points) are shown in Figure 11.3.

Parameter Learning

- Given some initialization we want to find the optimal:
  - μ_j Σ_j - Mean and covariance for each cluster
  - π_j Mixing Weights
- We cannot solve a closed form solution for this
- Need to use an iterative method that approaches local optimum
- We want to find a set of parameters θ that maximize log-likelihood
- Updating one parameter at a time and keeping the others fixed

Responsibilities

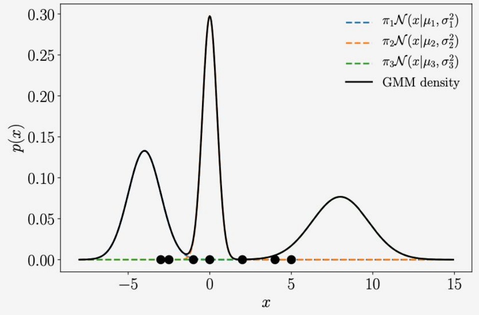
- We have k-components
- We can measure how much each point was described by each individually

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.057 & 0.943 & 0.0 \\ 0.001 & 0.999 & 0.0 \\ 0.0 & 0.066 & 0.934 \\ 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \in \mathbb{R}^{N \times K}$$

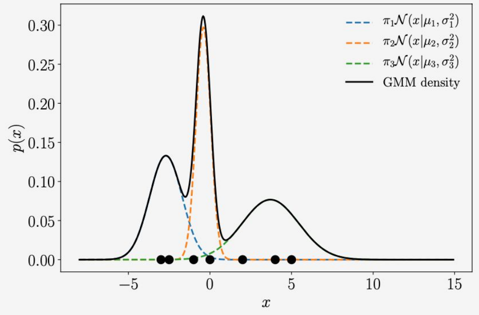
Updating the means

- Take a weighted average of the points based on responsibilities

$$\mu_k^{new} = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}}$$



(a) GMM density and individual components prior to updating the mean values.



(b) GMM density and individual components after updating the mean values.

In our example from Figure 11.3, the mean values are updated as follows:

$$\mu_1 : -4 \to -2.7 \quad (11.27)$$

$$\mu_2 : 0 \to -0.4 \quad (11.28)$$

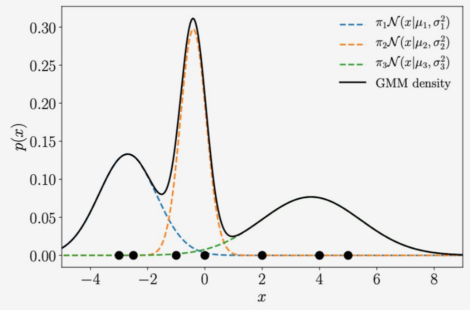
$$\mu_3 : 8 \to 3.7 \quad (11.29)$$

Updating the covariances

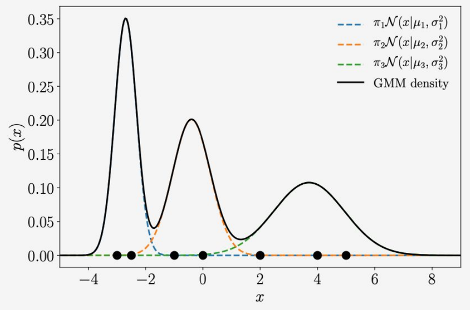
Weighted average of the covariances for the points each cluster is responsible for

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N} r_{nk}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top$$

$$\sigma_1^2 : 1 \rightarrow 0.14$$
$$\sigma_2^2 : 0.2 \rightarrow 0.44$$
$$\sigma_3^2 : 3 \rightarrow 1.53$$



(a) GMM density and individual components prior to updating the variances.

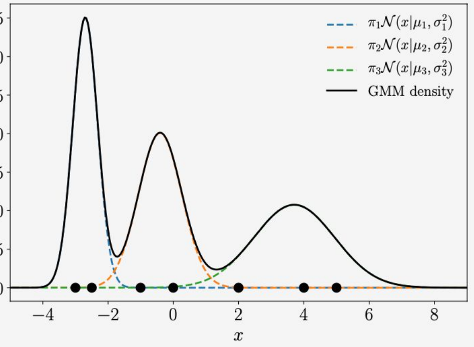


(b) GMM density and individual components after updating the variances.

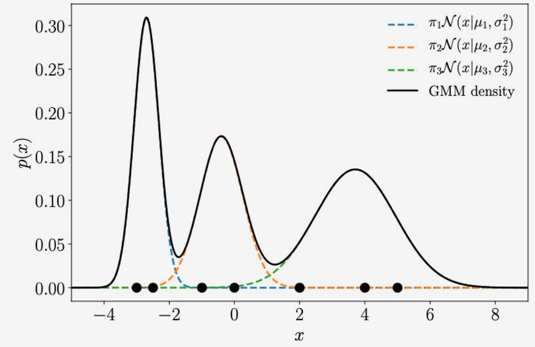
Updating the mixture weights

$$\pi_k^{new} = \frac{N_k}{N}, \quad k = 1, \ldots, K,$$

$$N_k := \sum_{n=1}^{N} r_{nk}$$



(a) GMM density and individual components prior to updating the mixture weights.

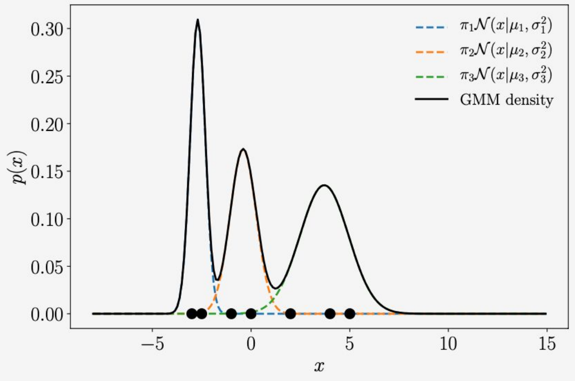


(b) GMM density and individual components after updating the mixture weights.

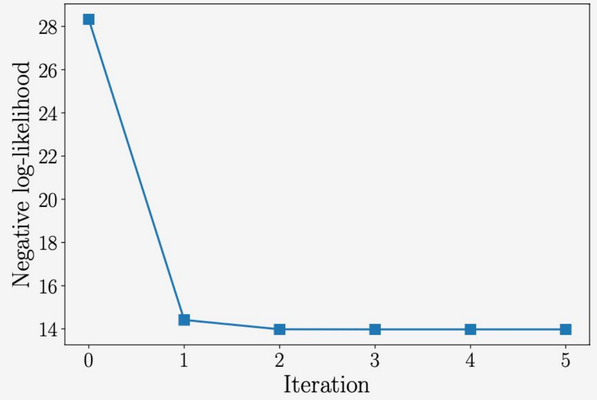
EM Algorithm

- E-Step - Expectation Step, measure responsibility
- M-Step, Maximization Step, update the parameters

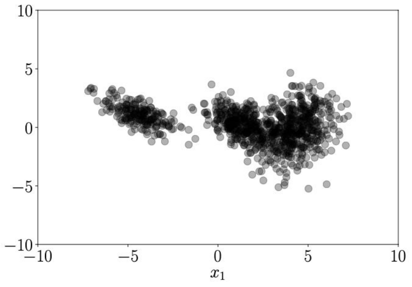
Example 11.6 (GMM Fit)



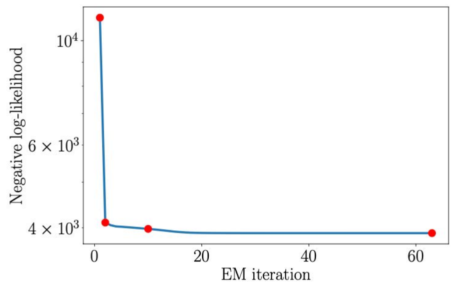
(a) Final GMM fit. After five iterations, the EM algorithm converges and returns this GMM.



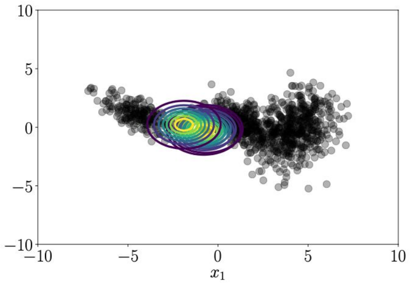
(b) Negative log-likelihood as a function of the EM iterations.



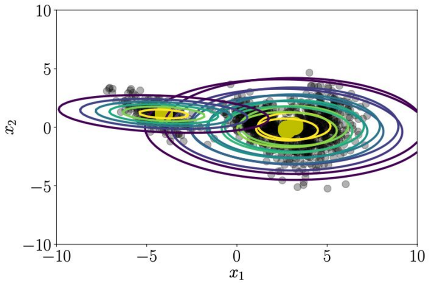
(a) Dataset.



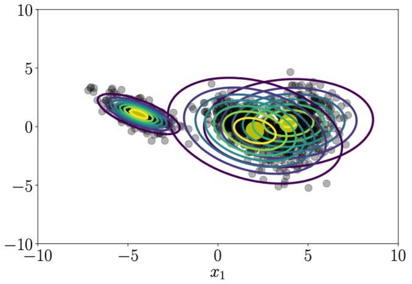
(b) Negative log-likelihood.



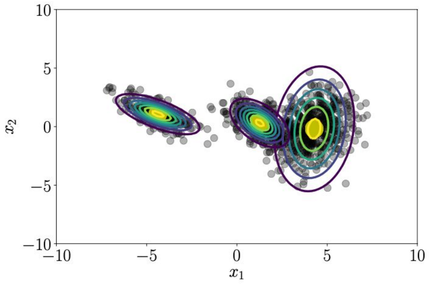
(c) EM initialization.



(d) EM after one iteration.



(e) EM after 10 iterations.



(f) EM after 62 iterations.

Latent-Variable Perspective

- We can think of GMMs in some way recovering latent variables
- Using the underlying means and covariances we can select a component and generate data from each distribution
- Imagine we have a dataset of animal weights, consisting of:
    - Lions
    - Rhinos
    - Humans
- We fit a GMM to this data, only capturing the weight
- The 3 components it pulls out will likely be decent approximations of each of these groups separately even though we never gave the model those indications explicitly
- We can generate data for each group