Gaussian Mixture Model

He He
Slides based on Lecture 13b from David Rosenberg's course materials
(https://github.com/davidrosenberg/mlcourse)

CDS, NYU

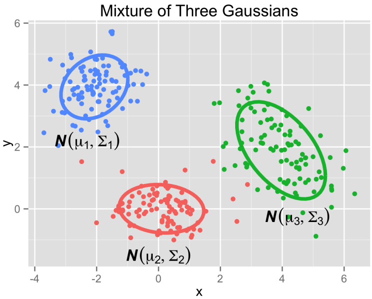
April 27, 2021

Probabilistic Model for Clustering

- Problem setup:
  - There are k clusters (or **mixture components**).
  - We have a probability distribution for each cluster.
- Generative story of a **mixture distribution**:
  1. Choose a random cluster $z \in \{1, 2, \ldots, k\}$.
  2. Choose a point from the distribution for cluster z.

Example:

1. Choose $z \in \{1, 2, 3\}$ with $p(1) = p(2) = p(3) = \frac{1}{3}$.
2. Choose $x \mid z \sim \mathcal{N}(X \mid \mu_z, \Sigma_z)$.



Gaussian mixture model (GMM)

Generative story of GMM with k mixture components:

1. Choose cluster $z \sim \text{Categorical}(\pi_1, \ldots, \pi_k)$.
2. Choose $x \mid z \sim \mathcal{N}(\mu_z, \Sigma_z)$.

Probability density of x:

- Sum over (marginalize) the **latent variable** z.

$$
\begin{aligned}
p(x) &= \sum_z p(x, z) && (1) \\
&= \sum_z p(x \mid z) p(z) && (2) \\
&= \sum_k \pi_k \mathcal{N}(\mu_k, \Sigma_k) && (3)
\end{aligned}
$$

Learning GMMs

How to learn the parameters π_k, μ_k, Σ_k?

- MLE (also called maximize marginal likelihood).
- Log likelihood of data:

$$
\begin{aligned}
L(\theta) &= \sum_{i=1}^{n} \log p(x_i; \theta) && (4) \\
&= \sum_{i=1}^{n} \log \sum_{z} p(x, z; \theta) && (5)
\end{aligned}
$$

- Cannot push log into the sum... z and x are coupled.
- No closed-form solution for GMM—try to compute the gradient yourself!

Learning GMMs: observable case

Suppose we observe cluster assignments z. Then MLE is easy:

$$n_z = \sum_{i=1}^{n} 1(z_i = z) \qquad \text{\# examples in each cluster} \tag{6}$$

$$\hat{\pi}(z) = \frac{n_z}{n} \qquad \text{fraction of examples in each cluster} \tag{7}$$

$$\hat{\mu}_z = \frac{1}{n_z} \sum_{i:z_i=z} x_i \qquad \text{empirical cluster mean} \tag{8}$$

$$\hat{\Sigma}_z = \frac{1}{n_z} \sum_{i:z_i=z} (x_i - \hat{\mu}_z)(x_i - \hat{\mu}_z)^T . \qquad \text{empirical cluster covariance} \tag{9}$$

Learning GMMs: inference

The inference problem: observe x, want to know z.

$$
\begin{align}
p(z=j \mid x_i) &= p(x, z=j)/p(x) \tag{10}\\
&= \frac{p(x \mid z=j)p(z=j)}{\sum_k p(x \mid z=k)p(z=k)} \tag{11}\\
&= \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_k \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)} \tag{12}
\end{align}
$$

- $p(z \mid x)$ is a *soft assignment*.
- If we know the parameters μ, Σ, π, this would be easy to compute.

EM for GMM

Let's compute the cluster assignments and the parameters iteratively.

The expectation-minimization (EM) algorithm:

1. Initialize parameters μ, Σ, π randomly.
2. Run until convergence:
    1. E-step: fill in latent variables by inference.
        - compute soft assignments $p(z \mid x_i)$ for all i.
    2. M-step: standard MLE for μ, Σ, π given "observed" variables.
        - Equivalent to MLE in the observable case on data weighted by $p(z \mid x_i)$.

M-step for GMM

- Let $p(z \mid x)$ be the soft assignments:

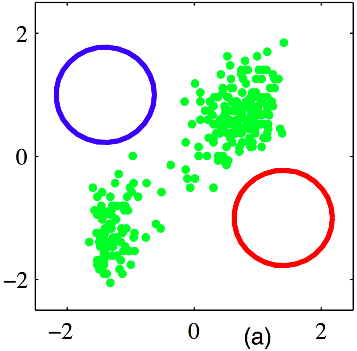
$$p(z_i = j \mid x_i) = \gamma_i^j = \frac{\pi_j^{\text{old}} \mathcal{N}\left(x_i \mid \mu_j^{\text{old}}, \Sigma_j^{\text{old}}\right)}{\sum_{c=1}^k \pi_c^{\text{old}} \mathcal{N}\left(x_i \mid \mu_c^{\text{old}}, \Sigma_c^{\text{old}}\right)}.$$

- Exercise: show that

$$\begin{aligned}
\mu_c^{\text{new}} &= \frac{1}{n_c} \sum_{i=1}^n \gamma_i^c x_i \\
\Sigma_c^{\text{new}} &= \frac{1}{n_c} \sum_{i=1}^n \gamma_i^c \left(x_i - \mu_c^{\text{new}}\right) \left(x_i - \mu_c^{\text{new}}\right)^T \\
\pi_c^{\text{new}} &= \frac{n_c}{n}.
\end{aligned}$$

EM for GMM

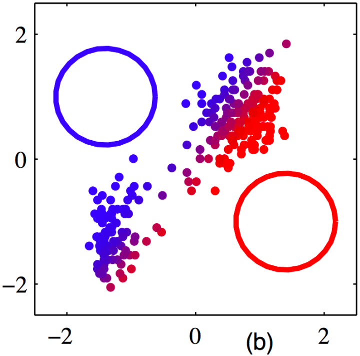
- Initialization



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

EM for GMM

- First soft assignment:

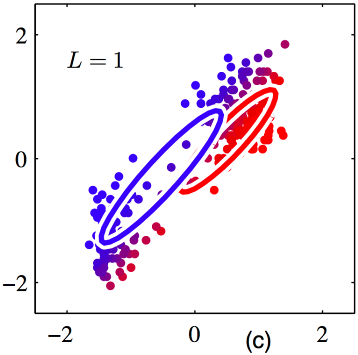


(b)

From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

EM for GMM

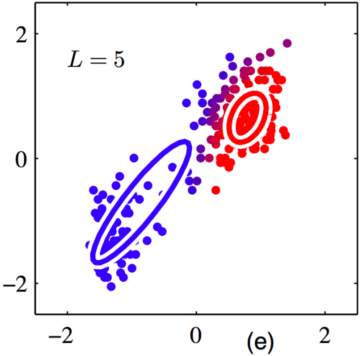
- First soft assignment:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

EM for GMM

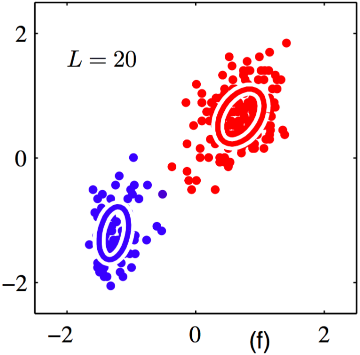
- After 5 rounds of EM:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

EM for GMM

- After 20 rounds of EM:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

EM for GMM: Summary

- EM is a general algorithm for learning latent variable models.
- *Key idea*: if data was fully observed, then MLE is easy.
    - E-step: fill in latent variables by computing $p(z \mid x, \theta)$.
    - M-step: standard MLE given fully observed data.
- Simpler and more efficient than gradient methods.
- Can prove that EM monotonically improves the likelihood and converges to a local minimum.
- k-means is a special case of EM for GMM with *hard assignments*, also called hard-EM.